ON SOME ASPECTS OF INTEGER LINEAR PROGRAMMING *

by

Rómulo H. González-Zubieta

AD 624553

Technical Report No. 16

On Work Performed under

Contract No. DA-31-124-ARO-D-209
U.S. Army Research Office (Durham)

Department of the Army Project No. 20011501B704

FUNDAMENTAL INVESTIGATIONS
in
METHODS OF OPERATIONS RESEARCH

Project D.S.R. 5217
Operations Research Center
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Cambridge 39, Massachusetts

June, 1965

20050222034

* - Adapted from a thesis presented to the Sloan School of Management in partial fulfillment of the requirements for the degree of Doctor of Philosophy, May 1965.

[illegible]

FOREWORD

The Center for Operations Research at the Massachusetts Institute of Technology is an inter-departmental activity devoted to graduate training and research in the field of operations research. Its products are books, journal articles, detailed reports such as this one, and students trained in the theory and practice of operations research.

The work of the Center is supported, in part, by government contracts and industrial grants-in-aid. Work reported herein was supported (in part) by the U. S. Army Research Office (Durham), under Contract Number DA-31-124-ARO-D-209. Reporduction in whole or in part is permitted for any purpose of the United States Government.

Philip M. Morse
Director of the Center

George R. Murray, Jr.
Research Associate

ACKNOWLEDGMENTS

I want to express my great appreciation to my thesis advisor, Professor John D. C. Little, for all his suggestions and encouragement, and in particular, for his help in proof-reading the manuscript.

I also want to thank Professor Martin Greenberger, Chairman of the Interdepartmental Doctoral Committee, and Professors Philip M. Morse and Ronald A. Howard, members of the same committee, for their contribution to my graduate education.

Dr. George R. Murray, Dr. Herbert P. Galliher, and Mrs. Nobu McNeill have been of great assistance to me during my four years as Research Assistant at the Operations Research Center.

The financial support of the Banco de Mexico, S. A. during the last two years, has made it possible for me to complete my graduate studies.

Thanks are due to Miss Carol Uhlinger for the excellent typing of the manuscript.

Finally, I want to express my gratitude to my wife for her reassuring support and for her constant optimism.

ON SOME ASPECTS OF INTEGER LINEAR PROGRAMMING

by

RÓMULO H. GONZÁLEZ-ZUBIETA

Submitted to the Sloan School of Management on May 25, 1965, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

ABSTRACT

A primal feasible (all-integer) integer linear programming algorithm has been developed and programmed, together with a related procedure for obtaining a first feasible solution. Once a feasible solution is found, the algorithm maintains feasibility at each stage, in contrast to other algorithms that have been programmed and are currently available. These other algorithms do not achieve feasibility until the optimal solution is reached. The primal feasible algorithm is based on a particular way of applying the cutting planes previously developed by R. E. GOMORY, and on a specific interpretation of their role.

The finiteness of convergence has been established for two-dimensional problems but not for the general case; however, there appears to be at least computational convergence in a considerable fraction of the cases.

In addition, a Generalized Euclidean Algorithm for finding the greatest common divisor for more than two numbers is defined. The solution of systems of linear diophantine equations is presented in terms of integer linear programming.

Some geometric considerations that help to illuminate the working of the algorithm, are examined.

Thesis Advisor: Dr. John D. C. Little
Title: Associate Professor of Industrial Management

TABLE OF CONTENTS

Pa

CHAPTER 1

INTRODUCTION

1.1 The Integer and the Mixed Linear Programming Problems

An Integer LP problem is a linear programming problem in which all the variables are constrained to take non-negative integer values. In a mixed LP problem some of the variables are constrained to be integer, while the rest are continuous. Both problems have in common that at least one of the variables is discrete. We can cover both problems by talking about LP with discrete variables, or discrete LP.

Discrete LP is important because there are many practical problems in which some of the variables are actually discrete; for instance, problems dealing with the number of flights, machines, men, or some other indivisible unit. If a problem of this type is attacked by solving the LP problem as non-discrete and by adjusting the solution thus obtained (so that it will fulfill the discreteness requirements), it might occur that the adjusted solution is far from optimal for the discrete problem. In many situations it will be profitable to use more refined methods so that the optimal solution is found.

Another reason why discrete LP is important is because logical relations may be expressed by means of Boolean variables (variables that can only take one of the values 0, 1), and thus some logical constraints can be incorporated in the definition of a problem. There are many combinatorial problems that can be formulated as discrete LP problems by using integer and Boolean Variables.

1.2 Available Integer LP Algorithms

In this section we give a brief description of the integer LP algorithms that are currently (May, 1965) available in the general literature of Operations Research and Management Science. Most of these algorithms use, in one way or another, the "cuts" originally discovered by R. E. GOMORY (refs. [4], [5] and [6]. These "Gomory-cuts" (or "Gomory inequalities") are secondary constraints that are implied by the set of original inequalities and by the discreteness of the variables. Gomory inequalities will be analyzed in the next chapter.

The first algorithm by Gomory [4], [5] uses the simplex technique to obtain the solution of the non-discrete problem. If any variable turns out to have a non-integer value a secondary constraint is generated. When this constraint is added it puts the problem in a "dual feasible, primal infeasible" form, and the dual simplex method is used to obtain a new optimal solution. If there are still some variables with non-integer variables, the procedure of generating a cut and re-solving is repeated until an all integer solution is found. The algorithm is shown to converge to the optimum in a finite number of steps.

The second of Gomory's algorithms [6] uses only integer coefficients to express all the inequalities and produces all-integer solutions throughout the process; but these solutions are not primal feasible because, together with the cuts, a dual simplex method is used. Thus, no feasible solution is obtained until the optimum is found. Convergence is assured if certain rules are followed.

The algorithm by GLENN T. MARTIN [8] is similar to the first of Gomory's algorithms, with the difference that the cuts that Martin generates are better (they go "deeper") and so he uses less iterations to get to the optimum. But as in the first case, no feasible all-integer solution is obtained until the end.

RICHARD VAN SLYKE AND ROGER WETS [11] have approached the problem of improving performance of integer LP algorithms by studying the way of easily generating efficient cutting planes. They handle this by using a triangular canonical form. A brief outline of their algorithm is given in their paper and the relationship with Gomory's algorithm [6] is examined.

FRED GLOVER has studied in [2] the effect of different heuristics upon the performance of Gomory's All-Integer Integer LP Algorithm [6]. In particular, he has developed the NOT (New Origin Technique) heuristic, which by means of the simplex algorithm provides a better starting point for Gomory's algorithm. In his second paper [3], Glover develops another approach for solving integer LP problems. An integer matrix is transformed into another integer matrix that exhibits certain substructure called bounding form. This substructure is operated upon and lower bounds are found for a subset of the variables. This information and the successively derived problem matrices are used in conjunction in order to guarantee finite convergence for the solution of the original problem. Again, no feasible solution is obtained until the end.

More recently, G. L. THOMPSON [10] has developed a solution enumeration approach that is characterized by: a) use of the number theoretic properties of the objective function to obtain a convenient transformation of variables, b) application of a multidimensional search procedure whose memory requirements vary only linearly with the size of the problem. Because of the way in which this search is conducted, a feasible solution is found only when an optimal solution is found. His method seems to be more efficient than Gomory's [6] ; furthermore, it can also be applied to mixed integer LP problems.

Thus no algorithm, of the ones presented so far, works on the principle of improving upon feasible integer solutions until the optimal solution is found. ADI BEN-ISRAEL and A. CHARNES [1] propose a Direct Algorithm that has this motivation. Their algorithm has in commor with ours the fact that they use the type of Gomory cuts which we call "non-zero natural inequalities". However, the Direct Algorithm has a very important shortcoming, which is that at certain stages an "Auxiliary Problem" has to be solved. This auxiliary problem is an integer LP problem that might be as large as the original problem; but they do not offer any systematic method for solving it, other than obtaining the solution of its continuous counterpart and "interpolating" from it an integer solution. The process of interpolating is not explained. The Direct Algorithm does not seem to be practical except for small problems; as a matter of fact, no computational results are reported.

1.3 Our Results

A) A primal feasible (all-integer) integer LP algorithm has been developed and programmed; it has the important characteristic that at any stage of the computation one always has a feasible integer solution. (Chapter 2)

B) A procedure for obtaining a first feasible solution has also been developed and programmed. In this formulalation we do not need additional variables or constraints as is usually necessary. Of special interest is a technique that takes advantage of any original equalities in order to reduce the size of the problem. (Chapter 3)

C) A proof of finite convergence of the primal algorithm, when applied to two-dimensional problems, has been found. (Chapter 4, section 4.2)

D) We have not established the finite convergence of the algorithm in the general case. Although this is disappointing, computational experience with other algorithms that are known to converge is not entirely satisfactory. A certain fraction of problems tried do not converge in any tolerable amount of time. From a practical point of view, then, the actual performance of an algorithm on the computer would appear to be its most important test. In this respect, the experience so far is that our algorithm is competitive with other algorithms. Because of the convergence difficulty that exists with all integer LP algorithms, the primal feasible property seems particularly desirable. There is no other programmed primal feasible algorithm that we know of. (Chapter 4, sections 4.3 and 4.4)

E) A class of Generalized Euclidean Algorithms, for finding the greatest common divisor of more than two numbers, is defined. Concepts of integer LP are applied to the problem of obtaining the general solution, and of generating non-negative solutions, of systems of linear diophantine equations. (Chapter 5)

F) Geometric considerations that throw light on the workings of the algorithm are examined. Of special interest is a technique (inverse pivoting) for expressing the relevant Gomory cuts explicitly in terms of the original variables at each stage of the algorithm. (Chapter 6)

G) Finally, some specific extensions are suggested. (Chapter 7)

CHAPTER 2

A PRIMAL FEASIBLE ALGORITHM

It is the purpose of the present chapter to develop a primal feasible algorithm for solving the integer LP problem. Before giving the outline of the basic algorithm we will introduce the necessary notation, present some basic properties of the tableau representation and discuss the special constraints called "GOMORY inequalities", which are essential for our algorithm.

2. 1 The Problem. Notation

Our notation will be similar to the one used by Gomory in references [5] and [6].

We will consider the problem of finding integers x_j $(j = 1, \ldots, n)$ that maximize

$$z = a_{0,0} + \sum_{j=1}^{n} a_{0,j}(-x_j) \qquad (2.1.1)$$

subject to

$$\sum_{j=1}^{n} a_{i,j}(x_j) \leq a_{i,0} \qquad i = 1,\ldots, m \qquad (2.1.2)$$

and to

$$x_j \geq 0 \qquad j = 1,\ldots, n \qquad (2.1.3)$$

where all the coefficients and constants are integers.

By introducing the non-negative slack variables s_i and by using the parametric variables t_j we can express the problem given by (2, 1. 1), (2. 1. 2) and (2. 1. 3) as follows:

Maximize z

with $$z = a_{0,0} + \sum_{j=1}^{n} a_{0,j}(-t_j) \qquad (2.1.4)$$

$$s_i = a_{i,0} + \sum_{j=1}^{n} a_{i,j}(-t_j) \qquad i = 1,\ldots, m$$

$$x_j = -1(-t_j) \qquad j = 1,\ldots, n$$

where the s_i and the x_j are non-negative integer variables, z is an unsigned integer variable, and all the coefficients and constants are integers.

In matrix notation we would have:

Maximize z

with $X = A^0 T^0$ $\qquad A^0 = (a_0, a_1, \ldots, a_n)$

$$X = \begin{bmatrix} z \\ s_1 \\ \vdots \\ s_m \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \qquad T^0 = \begin{bmatrix} 1 \\ -t_1 \\ -t_2 \\ \cdot \\ \cdot \\ \cdot \\ -t_n \end{bmatrix} \qquad (2.1.5)$$

where a_j is the j^{th} column of the $(m + n + 1)$ by $(n + 1)$ matrix formed with the constants and coefficients of (2. 1. 4), and where the same specifications apply.

The representation of the problem as given in (2. 1. 4) or in (2. 1. 5) is the parametric representation due to A. W. Tucker and it allows one to express every constraining variable and every variable "of interest" in terms of a set of non-basic or parametric variables t_j. (In the notation of (2. 1. 4), the s_i and the x_j are constraining variables -and also of interest- but z is not constraining, only of interest). Because of the way they have been defined, these parametric variables are also constrained to be non-negative and integer.

The relationships given in (2. 1. 4) can be expressed by means of a tableau of the following type:

$$\begin{array}{lllll}
 & 1 & -t_1 & & -t_n \\
\hline
z & = a_{0,0} & a_{0,1} & \cdots & a_{0,n} \\
s_1 & = a_{1,0} & & & \cdot \\
\cdot & \cdot & & & \\
\cdot & \cdot & & & \\
s_m & = a_{m,0} & \cdot\ \cdot & \cdot\ \cdot\ \cdot & \cdot\ \cdot \\
x_1 & = a_{m+1,0} & \cdot\ \cdot & \cdot\ \cdot & \cdot \\
\cdot & \cdot & & & \\
\cdot & \cdot & & & \\
x_n & = a_{m+n,0} & \cdot\ \cdot & \cdot\ \cdot & a_{m+n,n}
\end{array} \qquad (2.\ 1.\ 6)$$

2. 2 Pivoting. The Simplex Criterion

By using Gaussian elimination we can change the set of non-basic variables (i. e. the parametric variables), thus expressing the left hand variables in terms of different sets of parametric variables. In particular, if we want to exchange t_{j0} by s_{i0} we pivot on the element $a_{i0,\ j0}$. This means that we change the tableau according to the following formulas:

$$a'_{i,j0} = -(a_{i,j0}/a_{i0,j0}) \qquad i = 0,\ 1, \ldots,\ m+n$$

$$a'_{i,j} = a_{i,j} - a_{i0,j}\,(a_{i,j0}/a_{i0,j0}) \qquad j = 0,\ 1, \ldots,\ n;\ j \neq j0$$

(2. 2. 1)

This is equivalent to solving for t_{j0} in the $i0$th equation and replacing this result for t_{j0} into the other equations. (The marginal labels "s_{i0}" and "t_{j0}" exchange places on the outside of the tableau). In vector notation, formulas (2.2.1) become

$$\alpha'_{j0} = -\alpha_{j0}\,(1/a_{i0,j0})$$

$$\alpha'_{j} = \alpha_{j} - \alpha_{j0}\,(a_{i0,j}/a_{i0,j0}) \qquad j = 0, 1, \ldots, n;\ j \neq j0 \tag{2.2.2}$$

Our basic algorithm will always pivot on rows that will be added to the bottom of the tableau; so that the vertical set of marginal labels (i.e. z to x_n) will always remain the same.

The current solution is the vector of values that are obtained if every parametric variable is set equal to zero; in other words, it is the column vector $\alpha_0 = (a_{0,0}, a_{1,0}, \ldots, a_{m+n})$. The value of the solution is the value z, that is, $a_{0,0}$. A tableau is in primal feasible form (and the solution is said to be primal feasible, or simply, feasible) if every non-negative variable has a non-negative value, that is, if $a_{i,0} \geq 0$, $i = 1, \ldots, m+n$. A tableau is said to be in dual feasible form if $a_{0,j} \geq 0$, $j = 1, \ldots, n$ when we are maximizing z, or if $a_{0,j} \leq 0$, when we are minimizing z.

A column α_j is lexicographically positive $(\alpha_j \overset{\ell}{>} 0)$ if its first element, counting from the top down, is positive. Column α_j is (lexico-) larger than column α_i if $\alpha_j - \alpha_i \overset{\ell}{>} 0$. From now, when we refer to the sign of column, unless otherwise stated, it is to be understood in this lexicographical sense.

A tableau is said to be in *lexico-dual feasible form* if every column is positive (except, perhaps, column zero) when we are maximizing, or if every column is negative when we are minimizing.

If the tableau is both primal and dual feasible it means that the objective function has attained its *optimal* value and the current solution is one of the optimal solutions (see Appendix A). If the tableau is both primal feasible and lexico-dual feasible it means that the *lexico-optimal* solution has been found. That is, the lexico-largest feasible solution has been found if we are maximizing and the lexico-smallest if we are minimizing (see Appendix A). In either case we say that *optimality has been shown or proved*; in the latter case we also say that *lexico-optimality has been proved.*

The purpose of pivoting is to bring the tableau into an optimal form. We may start with a primal feasible form and try to attain dual feasibility by appropriate pivoting (while maintaining primal feasibility), or we may begin with a dual feasible form and try to attain primal feasibility (while maintaining dual feasibility). The first technique is usually called a (primal) simplex method, and the second one a dual simplex method. If instead of trying to attain, or to maintain, plain dual feasibility, we try to attain, or to maintain, lexico-dual feasibility, we will have a lexicographical simplex method, and a lexicographical dual simplex method respectively.

In Appendix A we present an example that shows that it is necessary to use a lexicographical simplex method in our basic algorithm, rather than a "plain" simplex method. Whenever we mention the simplex method in the rest of this paper, we will be referring to the lexicographical simplex method, unless otherwise stated. Also, we will usually assume that the objective is to maximize.

If the pivoting element is $a_{i0,j0}$, row i0 is called the pivoting row and column j0 is the pivoting column (column zero is not considered for pivoting).

When using the simplex method we will pivot on negative columns if we are maximizing, and we will pivot on positive columns if minimizing. The reason for this will be clear below.

Assume we have chosen a column, j0, for pivoting; the simplex method criterion for choosing a pivoting row i0, is then the following:

Let I be the set of indices i, $i \neq 0$, such that $a_{i,j0} > 0$. Choose i0 so that:

$$(a_{i0,0}/a_{i0,j0}) = \min_{i \in I} \{a_{i,0}/a_{i,j0}\} \tag{2.2.3}$$

If the set I is empty and we are minimizing, it means that the objective function is unbounded. If we are maximizing, it either means that the objective function is unbounded (if $a_{0,j0} < 0$), and/or that some of the problem variables are unbounded (for further discussion see section 4.1 and Appendix A).

The simplex criterion for pivoting guarantees that the solution will stay primal feasible if it is already primal feasible.

If we are maximizing, the column vector α_0 will be (lexicographically) non-decreasing because $a_{i0,0} \geq 0$, $a_{i0,j0} > 0$, $\alpha_{j0} \overset{\ell}{<} 0$ and from (2.2.2)

$$\alpha'_0 = \alpha_0 - \alpha_{j0}(a_{i0,0}/a_{i0,j0}) \overset{\ell}{\geq} \alpha_0$$

If we were minimizing we would choose $\alpha_{j0} \overset{\ell}{>} 0$ and α_0 would be non-increasing .

For further details on the basic theory of linear programming see any standard textbook such as, Hadley, G., Linear Programming, Reading, Mass., Addison-Wesley, 1961.

2.3 Gomory Inequalities

As mentioned before, most of our results make use of the Gomory Inequalities (or Gomory cuts), which were first developed by Gomory and now bear his name. In order to make our exposition more self-contained we will reproduce here, in a slightly modified form, the proof of Ben-Israel and Charnes for the Gomory Inequalities (from [1]).

Ben-Israel and Charnes Theorem

For any real number x let

$[x]$ denote the largest integer $\leqslant x$

$\langle x \rangle$ denote the smallest integer $\geqslant x$

Also, let L denote the set of integer vectors in E^n and E^n_+ the set of non-negative vectors in E^n.

Ben-Israel considers inequalities of the type

$$\sum_{j=1}^{n} a_i x_i \leqslant a_0 \qquad (2.3.1)$$

and shows that integer vectors $x = (x_1, \ldots, x_n)$ which satisfy (2.3.1) lie in a closed half-space which depends on (2.3.1) but always contains E^n_+, satisfy other constraints which will be called Gomory Inequalities.

For this, the coefficients in (2.3.1) are decomposed, for any real number $p > 0$, into their integral and fractional parts as follows:

$$a_i = p\,[a_i/p] + f_i \qquad i = 0, 1, \ldots, n \qquad (2.3.2)$$

where $0 \leqslant f_i < p$

Theorem (part [a]):

Let p be any positive real number and f_i as defined in (2.3.2)

If $x \in L$ satisfies (2.3.1) and

$$\sum_{i=1}^{n} f_i x_i \geq 0 \tag{2.3.3}$$

then

$$\sum_{i=1}^{n} [a_i/p] \; x_i \leq [a_0/p] \tag{2.3.4}$$

<u>Proof</u>

Rewrite (2.3.1) as

$$p \sum_{i=1}^{n} [a_i/p] \; x_i + \sum_{i=1}^{n} f_i x_i \leq p \; [a_0/p] + f_0 \tag{2.3.5}$$

now there are two possible cases:

Either

$$- \sum_{i=1}^{n} f_i x_i \leq - f_0 \tag{2.3.6}$$

in which case (2.3.4) always holds regardless whether $x \in L$ or not, or else

$$\sum_{i=1}^{n} f_i x_i < f_0 \tag{2.3.7}$$

in which case we use the following inequality which always follows from (2.3.5), upon dividing by p, and taking integer parts on both sides:

$$\sum_{i=1}^{n} [a_i/p]\, x_i + (1/p) \sum_{i=1}^{n} f_i\, x_i \leq [a_0/p] + f_0/p \qquad (2.3.8)$$

Now, since $x \in L$, $\sum_{i=1}^{n} [a_i/p]\, x_i$ is an integer and, using (2.3.3), (2.3.7), and the fact that $0 \leq f_0 < p$, we finally obtain (2.3.4). This proves part [a] of Ben-Israel and Charnes Theorem.

The second part of his theorem is actually a corollary of part [a] as we will presently show.

Part [b] states

Let p be any positive real number, and let g_i be defined by

$$b_i = p \langle b_i/p \rangle - g_i \qquad i = 0, 1, \ldots, n \qquad (2.3.9)$$

if $x \in L$ satisfies

$$\sum_{i=1}^{n} b_i\, x_i \geq b_0 \qquad (2.3.10)$$

and

$$\sum_{i=1}^{n} g_i\, x_i \geq 0 \qquad (2.3.11)$$

then

$$\sum_{i=1}^{n} \langle b_i/p \rangle \; x_i \geq \langle b_0/p \rangle \qquad (2.3.12)$$

We state that part [b] is a corollary of part [a]

Proof

Let us define b_i as

$$b_i = - a_i \qquad i = 0, 1, \ldots, n \qquad (2.3.13)$$

We will make use of the fact that for any real number c this holds:

$$- [c] = \langle -c \rangle \text{ and } [-c] = - \langle c \rangle \qquad (2.3.14)$$

We can rewrite (2.3.2) as follows

$$-b_i = p [-b_i/p] + f_i \qquad i = 0, \ldots, n \qquad (2.3.15)$$

then, by (2.3.14),

$$b_i = p \langle b_i/p \rangle - f_i \qquad (2.3.16)$$

we see that (2.3.9) is met if we define

$$g_i = f_i \qquad i = 0, \ldots, n \qquad (2.3.17)$$

Relationship (2.3.1) may be rewritten as

$$\sum_{i=1}^{n} (-b_i) \, x_i \leq -b_0 \qquad (2.3.18)$$

from which (2. 3. 10) follows.

Also, (2. 3. 11) follows from (2. 3. 3) and (2. 3. 17).

Finally, (2. 3. 12) is obtained from (2. 3. 4) by rewriting it as

$$\sum_{i=1}^{n} [-b_i/p] \; x_i \leq [-b_0/p]$$

and, by (2. 3. 14)

$$-\sum_{i=1}^{n} \langle b_i/p \rangle \; x_i \leq - \langle b_0/p \rangle$$

We have thus proved that part [b] of the theorem is implied by part [a] . (The converse is also true).

Part [a] of the theorem is due to Gomory [6] but instead of condition (2. 3. 3) he has the stronger condition that $x \in E^n_+$. He proves the theorem for the cases $p = 1$, and $p > 1$.

Thus, we have proved that any Gomory cut generated from some constraint s is satisfied by every non-negative integer vector x that satisfies s . Whenever we have an integer LP problem defined by a set of constraints we could add any number of Gomory cuts as generated from the original inequalities, and the expanded set of inequalities would still be feasible for (satisfied by) the same set of non-negative integer vectors.

Gomory Cuts in Tableau Notation

In the tableau, the ith row is

$$s_i = a_{i,0} + \sum_{j=1}^{n} a_{i,j}(-t_j) \; ; \; s_i \geq 0$$

and is equivalent to

$$\sum_{j=1}^{n} a_{i,j} t_j \leq a_{i,0}$$

its Gomory cuts will be:

$$\sum_{j=1}^{n} a_{i,j}/p \quad t_j \leq a_{0,0}/p \; ; \; p > 0$$

or

$$\sum_{j=1}^{n} b_j t_j \leq b_0 \qquad (2.3.19)$$

where

$$b_j = a_{i,j}/p \qquad j = 0, 1, \ldots, n \qquad (2.3.20)$$

All the coefficients and variables of (2.3.19) are integers so that if we introduce a slack variable r, it will be both non-negative and integral

$$r = b_0 + \sum_{j=1}^{n} b_j (-t_j) \tag{2.3.21}$$

The cut (2.3.19) or its equivalent (2.3.21) may be added to the original constraints by adding to the tableau another row with the coefficient $(b_0, b_1, \ldots, b_n)$.

If this added row is used as the pivot row, the pivoting formulas (2.2.1) will be changed into:

$$a'_{j0} = - a_{j0} (1/b_{j0})$$

$$a'_j = a_n - a_{j0} (b_j/b_{j0}) \qquad j = 0, \ldots, n; \quad j \neq j0 \tag{2.3.22}$$

A Gomory cut will be called a <u>non-zero-cut</u> if $b_0 \neq 0$ and a <u>zero-cut</u> if $b_0 = 0$.

In the next section and in the following chapters we will study further the theory and the application of Gomory cuts.

2.4 Development of the Basic Algorithm

We want to develop an "All-Integer Primal Feasible Integer LP Algorithm". This means that we want the current solution to always be (primal) feasible and integral; furthermore, we want all the coefficients to remain integral.

If the starting solution is feasible and all the coefficients are integer, feasibility can be reserved by using the simplex criterion for pivoting; and integrality of the coefficients can be maintained by having the pivoting elements equal to 1 (see (2.2.1)).

In general, it is not always possible to find pivoting elements that satisfy the simplex criterion and that have, at the same time, a value of 1 . However, such a pivoting element may always be obtained by properly generating a Gomory cut and appending it to the tableau, as will be shown below.

Row i0 is <u>pivotable on column j0</u> if $a_{i0,j0} > 0$. Let I_{j0} be the set of rows that are pivotable on column j0 . A pivotable row i0 is <u>most-binding</u> on column j0 if it satisfies the simplex-like criterion:

$$[a_{i0,0}/a_{i0,j}] = \min_{i \in I_{j0}} \{[a_{i,0}/a_{i,j0}]\} \qquad (2.4.1)$$

If $a_{j0} < 0$ and if row i0 is most-binding on column j0 , we may generate a cut from row i0 by letting $p = a_{i0,j0}$. This cut may be appended to the tableau and its element b_{j0} may be used as the pivoting element. (Row i0 will be called the <u>generating row</u>, and $a_{i0,j0}$ will be the <u>generating element</u>). The generated cut will not only satisfy (2.4.1), but it will also satisfy the simplex criterion (2.2.3). Furthermore, the pivoting element b_{j0} will be equal to 1 (as may be checked from its definition (2.3.20)), and the pivoting formulas (2.3.20) may be simplified

$$a'_{j0} = -a_{j0}$$

$$a'_j = a_j - a_{j0}\, b_j \qquad j = 0, \ldots, n\,;\; j \neq j0 \qquad (2.4.$$

These new formulas show that the integrality of the tableau elemen will be preserved upon pivoting.

If some row i is pivotable on column j and it has $a_{i,j} > a_{i,0}$ it will have $[a_{i,0}/a_{i,j}] = 0$ and it is clear that it will be most-binding on column j. Even more, we say it is zero-binding because a zero-cut coul be generated from it.

Considering column j, if there is at least one pivotable row and if none of the pivotable rows is zero-binding we have a breakthrough becaus we can generate a non-zero-cut (the breakthrough is in the maximizing di tion if $a_j < 0$). If there is no pivotable row on column j (and if $a_j < 0$) th the problem has an unbounded solution (which is another type of breakthro

The Basic Algorithm (Maximizing Version)

A - If there is a breakthrough in a negative column $j0$, select suc a column for pivoting. If no negative column has a breakthrough take som negative column $j0$ for pivoting. (If there are no negative columns the tableau is optimal).

B - Having chosen $j0$, take one of the most-binding rows on colum

$j0$ as the generating row $i0$ (if there is none, the problem is unbounded).

C - From row $i0$ generate a Gomory cut with $p = a_{i0,j0}$ and add this constraint to the bottom of the tableau.

D - Pivot on element b_{j0} of the added constraint and erase the constraint after having pivoted. Go back to A.

The basic algorithm assumes that the tableau is in primal feasible form and that all the coefficients are integer. Also, it does not specify how to choose a pivoting column from several proper candidates, or how to choose a generating row when there are several most-binding rows on the pivoting column $j0$. These matters will be discussed in Chapter 4. If we want the algorithm to be minimizing we only have to require that the pivoting columns be positive.

In the process of solution there will be an improvement in the solution vector α_0 whenever a breakthrough is found and used. Going from one breakthrough to another constitutes a major iteration. But in order to obtain a breakthrough it might be necessary to perform several pivots with zero-cuts. Every pivot, whether it is done with a zero-cut or with a non-zero-cut, is a minor iteration (or simply, iteration). Thus, a major iteration requires a number of minor iterations.

2.5 An Example

We will solve here the example given by Martin in [8]

$$\max\ z = 2x_1 + 3x_2$$

$$\text{with} \qquad 2x_1 + 5x_2 \leq 8$$

$$3x_1 + 2x_2 < 9$$

Introducing the slack variables and putting the problem into the tableau form:

$$\begin{array}{cc|ccc|cc}
 & & 1 & -x_1 & -x_2 & & \\
\hline
z & = & 0 & -2 & -3 & & \\
s_1 & = & 8 & 2 & 5^{\circ} & \leftarrow\ \leftarrow & \\
s_2 & = & 9 & 3 & 2 & & p = 5 \\
x_1 & = & 0 & -1 & 0 & & \\
x_2 & = & 0 & 0 & -1 & & \\
\hline
 & & & & & & \\
r_1 & = & 1 & 0 & 1^{*} & & \\
 & & & & \uparrow & &
\end{array}$$

The arrows show the column j0 and the row i0 . The dot (o) marks the element $a_{i0,\,j0}$ and the asterisk (*) marks the pivoting element in the added constraint.

Pivoting and repeating the process:

$$\begin{array}{cc|ccc|c}
 & & 1 & -x_1 & -r_1 & \\
\hline
z & = & 3 & -2 & 3 & \\
s_1 & = & 3 & 2^{\circ} & -5 & \leftarrow \\
s_2 & = & 7 & 3 & -2 & \\
x_1 & = & 0 & -1 & 0 & \\
x_2 & = & 1 & 0 & 1 & \\
\hline
r_2 & = & 1 & 1^{*} & -3 & \\
 & & & \uparrow & &
\end{array}
\quad
\begin{array}{cc|ccc|c}
 & & 1 & -r_2 & -r_1 & \\
\hline
z & = & 5 & 2 & -3 & \\
s_1 & = & 1 & -2 & 1 & \\
s_2 & = & 4 & -3 & 7^{\circ} & \leftarrow \\
x_1 & = & 1 & 1 & -3 & \\
x_2 & = & 1 & 0 & 1 & \\
\hline
r_3 & = & 0 & -1 & 1^{*} & \\
 & & & & \uparrow &
\end{array}$$

$$\begin{array}{cc|ccc|c}
 & & 1 & -r_2 & -r_3 & \\
\hline
z & = & 5 & -1 & 3 & \\
s_1 & = & 1 & -1 & -1 & \\
s_2 & = & 4 & 4 & -7 & \\
x_1 & = & 1 & -2 & 3 & \\
x_2 & = & 1 & 1^{\circ} & -1 & \leftarrow \\
\hline
r_4 & = & 1 & 1^{*} & -1 &
\end{array}
\quad
\begin{array}{cc|ccc|c}
 & & 1 & -r_4 & -r_3 & \\
\hline
z & = & 6 & 1 & 2 & \leftarrow \\
s_1 & = & 2 & 1 & -2 & \text{Op} \\
s_2 & = & 0 & -4 & -3 & \\
x_1 & = & 3 & 2 & 1 & \\
x_2 & = & 0 & -1 & 0 &
\end{array}$$

The optimal solution is $x_1 = 3,\ x_2 = 0,\ z = 6$.

CHAPTER 3

OBTAINING A FIRST FEASIBLE SOLUTION

3.1 Different Types of Variables

Every row of the tableau corresponds to an expression that gives a certain variable in terms of the (current) parametric variables. The variables thus expressed will be called row variables and they include the objective, the slack variables, and the problem variables. The constraints of the original integer LP are denoted by the fact that some (or all) of these row variables are constrained. Four types of variables will be studied: unsigned variables, non-negative variables, zero-variables and bounded variables. In addition, the concept of objective variable will be discussed.

3.1.1 Unsigned Variables

An unsigned variable is a variable that is not, *per se*, necessarily constrained in sign. The most common case of unsigned variable is the one corresponding to the objective function, but in some problems there might be additional unsigned variables that are pertinent to the formulation of the problem. The treatment of rows corresponding to unsigned variables is simple; they are not considered as candidates for generating cuts, but the corresponding coefficients are transformed by means of the pivoting formulas (2.4.3) as are those of any other row. Unsigned variables are always considered to be feasible, of course.

3.1.2 Non-negative Variables

A non-negative variable may be of any of two types: a slack variabl that was introduced to change an inequality into an equality, or an origina problem variable that was required to be non-negative. Non-positive variables are not considered here because they can be converted to non-negative variables by a simple change of variable. A non-negative variable s_i is feasible if its current value, a_{i0}, is non-negative. The row corresponding to a non-negative variable will be a candidate for generating cuts only if it is feasible (for an exception see section 3.2).

3.1.3 Zero-Variables (The Treatment of Equalities)

One way to deal with equalities in linear programming is to represent them by means of opposing inequalities (for instance, $ax = b$, $cx = d$ may be represented as $ax \leq b$, $cx \leq d$, $-(a + c)x \leq -(b + d)$). As far as we know, this has been the usual approach in integer LP[1].

We have used a different approach.

1 See, for instance, Giglio, R. J. and Wagner, H. M. "Approximate Solutions to the Three-Machine Scheduling Problem", Operations Research, v. 12 (1964), pp. 305 - 324.

Take the following equality:

$$\sum_{i=1}^{n} a_i x_i = a_0 \qquad (3.1.1)$$

Let us introduce an artificial variable s and put (3.1.1) in the following form:

$$s = a_0 + \sum_{i=1}^{n} a_i (-x_i) \qquad (3.1.2)$$

It is obvious that any set of x_i's that satisfy (3.1.1) (that is, any set of feasible x_i's), will make s equal to zero. We call s a <u>zero-variable</u> because it is constrained to be zero; that is, s is <u>feasible only when its value is zero</u>. Furthermore, a variable of this type is an <u>explicit</u> zero-variable since we explicitly know its nature from the problem formulation.

If, after several pivots, a zero-variable row has only one non-zero element $a_{i,j}$, $j \neq 0$, this means that $s_i = a_{i,j}(-t_j)$, so that the only feasible value of the parametric variable t_j is zero. If this is the case we might as well make $t_j = 0$ which is equivalent to erasing the column corresponding to t_j, thus <u>reducing</u> the width of the tableau. Furthermore, if after erasing a column there are one or more rows that have only zero elements, (except perhaps in column zero), we may also erase these rows if we take care of storing somewhere the current (and final) values of the corresponding row variables. The justification for this is that if every element $a_{i,j}$, $j \neq 0$, of row i is equal to zero, then there is no way in which $a_{i,0}$ could change by pivoting (we would always have $a_{i,j0} = 0$ in the pivoting formulas (2.2.1)).

This reduction of the tableau is of course equivalent to using the equality in question to solve for one variable in terms of the others, and to substitute this expression in the other constraints so as to reduce the dimensionality of the problem. The algorithm for finding a first feasible solution will do this for us, automatically.

But this is not the only way to reduce the dimensionality of the problem. Because of the special nature of integer LP problems, even if there are no equalities in the formulation of the problem, in many cases the dimensionality of the set of feasible integer solutions will be smaller than the dimensionality of the space of feasible continuous solutions. Thus, in the process of solution of a problem we may find out that some of the parametric variables become, in fact, implicit zero-variables. If this situation occurs and we are able to recognize it, the tableau can be reduced. The following lemma gives the basic rule for recognizing implicit zero variables.

Lemma 3.1: If there is a row i such that $a_{i,j} \geqslant 0$, $j = 0$, 1, . . . , n , and this row corresponds to a non-negative variable, then every column j with $a_{i,j} > a_{i,0}$ corresponds to an implicit zero-variable.

<u>Proof</u>: Let $J^- = \{j \mid 0 < a_{i,j} \leq a_{i,0} ; j = 1, \ldots, n\}$ and $J^+ = \{j \mid a_{i,j} > a_{i,0} ; j = 1, \ldots, n\}$. Let us assume that J^+ is not empty so that the lemma applies. Because row i corresponds to a non-negative variable the row represents the following constraint

$$\sum_{j \in J^-} a_{i,j} t_j + \sum_{j \in J^+} a_{i,j} t_j \leq a_{i,0}$$

A cut can be generated by letting $p = \min_{j \in J^+} \{a_{i,j}\}$ the coefficients of the cut will be:

$$b_0 = [a_{i,0}/p] = 0$$

$$b_j = [a_{i,j}/p] \quad \begin{array}{l} \geq 1 \text{ if } j \in J^+ \\ = 0 \text{ if } j \in J^- \end{array}$$

The Gomory cut is then: $\sum_{j \in J^+} b_j t_j \leq b_0 = 0$ with $b_j \geq 1$ for $j \in J^+$.

But since all the parametric variables t_j can be constrained to be non-negative (without losing feasible solutions to the original problem) then the only feasible value for the variables t_j , $j \in J^+$ is zero. Thus, $t_j = 0$, $j \in J^+$, implies that this set of parametric variables are zero-variables and their corresponding column may be erased in order to reduce the tableau. (Q.E.I

A final remark about zero-variables: a zero-variable is at the same time a non-negative and a non-positive variable, and we may consider it so whenever it is convenient.

3.1.4 Bounded Variables

Many of the practical problems formulated as integer LP problems, involve the use of variables that are constrained to be either zero or one; these variables are sometimes called Boolean or decision variables. Generalizing this concept, let us assume that the integer variable x_i is bounded as follows $0 \leq x_i \leq M_i$ (where M_i is some positive integer). This requirement could be expressed with two separate rows of the tableau one row would express $x_i \geq 0$ and the other one would stand for $x_i^s = M_i - x_i \geq 0$.

But it is possible to express both constraints with a single row (this saving computer memory). If we have the constraint

$$x_i = a_{i,0} + \sum_{j=1}^{n} a_{i,j} (-t_j) \geq 0$$

we can <u>compute</u> the <u>counter-constraint</u>

$$x_i^s = (M_i - a_{i,0}) + \sum_{j=1}^{n} (-a_{i,j}) (-t_j) \geq 0$$

Thus, the slack variable x_i^s is feasible if $M_i - a_{i,0} \geq 0$. When x_i^s is feasible, it will be binding on column j if $a_{i,j} < 0$; whereas x_i, when feasible, will be binding on column j if $a_{i,j} > 0$.

In order to handle a bounded variable it is enough to state its non-negativity, the value M_i, and the fact that the variable is bounded (an arbitrary label would be used to specify this in the computer program). For analytical purposes we will think of it as two separate non-negative variables.

3.1.5 Objective Variables

Any variable, whether unsigned, non-negative or zero-variable, can be considered to be the problem objective at any stage of the process of solution of a given problem. The motivation for doing this will be discussed in the next section.

If we are taking s_i as the current objective function we will consider its row as the zeroth row, that is, as the topmost row. (The order of the other rows should be specifically assigned, but the criterion for assigning the order is unimportant, as long as the rows retain their ordering). If s_i is being maximized, only lexico-negative columns will be considered proper candidates for pivoting. If there is no negative column it means that s_i cannot be maximized further. (If s_i is being minimized only positive columns are considered for pivoting).

3.2 Finding a First Feasible Solution

Before proceeding to optimize the original objective function, we require that the tableau be in a (primal) feasible form. There are two types of infeasibilities that we want to remove: a) zero-variables with non-zero values, b) non-negative variables with negative values. It seems advisable to make the zero-variables feasible first so that we may reduce the size of the tableau as soon as possible.

One way to deal with zero-variables that are not feasible is the following: Let us assume that s_i is a zero-variable and its value is $a_{i,0} > 0$ (if $a_{i,0} < 0$ we could multiply the row by -1 and work with s_i' ($s_i' = -s_i$) which is also a zero variable). We will consider the problem defined only by those rows corresponding to non-negative variables that are already feasible, and we will try to minimize s_i subject to $s_i \geq 0$, that is, considering row i itself as a constraint. To do this we apply our basic algorithm to the subproblem; that is, we pivot in such a way that the non-negativity of the subproblem variables is preserved while we seek to bring the value of s_i down to zero. Thus, we will pivot only in columns that are lexico-positive in the subproblem. If there is no such a column for the row i under consideration, and if $a_{i,0} > 0$, this means that no feasible solution exists for the original problem. (The solution space of the original problem is contained within the solution space of the subproblem; hence, if the subproblem does not have any feasible solution neither does the original problem).

If we want to reduce the tableau by eliminating the zero-variable we will not be satisfied with driving its value down to zero. We have to continue pivoting in such a way that row i will eventually have only one non-zero element. This will be accomplished in a finite number of steps as will be shown next.

Lemma 3.2: If we generate a Gomcry cut from row $i0$, using $p = |a_{i0,j0}| \neq 0$, and we pivot on column $j0$ of the cut, the new elements of row $i0$ will satisfy the following: $a'_{i0,j0} = -|a_{i0,j0}|$; $0 \leq a'_{i0,j} < |a_{i0,j0}|$, $j \neq j0$. That is, every element of row $i0$ will be non-negative and smaller than $|a_{i0,j0}|$; except the element on column $j0$, which will be negative and equal to $-|a_{i0,j0}|$.

Proof: We have defined $[c]$ to denote the largest integer smaller than, or equal to c. Thus, if $b \neq 0$, the following holds:

$$0 \leq (a/b - [a/b]) < 1$$

If $b > 0$ we have

$$0 \leq (a - b[a/b]) < b$$

or

$$0 \leq (a|b) < b \qquad (3.2.1)$$

where $\qquad (a|b) \equiv a - b[a/b], \; b > 0 \qquad (3.2.2)$

This property will be used to prove the lemma.

The cut generated from row i0 with $p = | a_{i0,j0} |$ will have the following coefficients:

$$b_j = [a_{i0,j} / | a_{i0,j0} |] \qquad j = 0, \ldots, n$$

There may be two cases: $a_{i0,j0} > 0$, or $a_{i0,j0} < 0$. If $a_{i0,j0} > 0$, the pivoting formulas will be those of (2.4.3), and specifically for $i = i0$, we will have:

$$a \quad) = - a_{i0,j0} = - | a_{i0,j0} |$$

$$a'_{i0,j} = a_{i0,j} - a_{i0,j0} b_i = a_{i0,j} - | a_{i0,j0} | \ [a_{i0,j} / | a_{i0,j0} |]$$

$$j = 0, \ldots, n \qquad j \neq 10$$

If $a_{i0,j0} < 0$, we will have $b_{j0} = -1$, and $a_{i0,j0} = - | a_{i0,j0} |$. We can obtain the pivoting formulas for row i0 from (2.3.22), that is:

$$a'_{i0,j0} = + a_{i0,j0} = - | a_{i0,j0} |$$

$$a'_{i0,j} = a_{i0,j} + a_{i0,j0} b_j = a_{i0,j} - | a_{i0,j0} | \ [a_{i0,j} / | a_{i0,j0} |]$$

$$j = 0, \ldots, n \qquad j \neq j0$$

Thus, in either case we have the same pivoting formulas for row i0, and by comparing them with (3.2.1), we see that Lemma 1 is indeed satisfied. (Q.E.D.

<u>Lemma 3.3</u>: If we have a row i with $a_{i0} = 0$, we can reduce it.

in a finite number of steps, so that only one non-zero element will remain.

Proof: The way to obtain this reduction is to use only positive elements of row i as generating elements. (If initially there are no positive elements in row i, we may use any of the negative elements as generating elements. From Lemma 3.2 we see that the other elements of row i will become either positive or zero. If at least one of them becomes positive we may apply the rule given above. If they all become zero then Lemma 3.3 is already satisfied.)

By always taking positive generating elements from row i, we are assured by Lemma 3.2 that they will always be monotonically decreasing. But they cannot decrease forever because they are positive and they decrease by integer amounts. Thus, after a finite number of iterations there will be no more positive elements in row i. The only way this may happen is if every element is zero except the one that was used last as generating element, as may be checked from Lemma 3.2. This proves Lemma 3.3. (Q.E.D.)

The procedure for row reduction will be accelerated if the smallest of the positive elements is chosen each time as the generating element.

When the row of a zero-variable is reduced so that it has only one non-zero element a_{ij}, $j \neq 0$, we know that the parametric variable associated with column j is a zero-variable, and we may erase this column, thus reducing the size of the tableau.

Once we have eliminated one zero-variable we take another one and repeat the procedure until there are no infeasible zero-variables. Then we may turn our attention to removing the infeasibility of non-negative variables, if there is any infeasibility of this type.

We have said that in removing the infeasibility of one zero-variable the pivoting is done in such a way that the variables that are already feasible, remain feasible. This would, perforce, slow down the process of removing that infeasibility and of reducing the tableau. Another strategy would be to concentrate [illegible] reducing the elements of the row and in eliminating the zero-variable, without worrying whether any other variables become infeasible. This would bring about a faster reduction of the tableau, and once it has been reduced it might be more efficient to worry about the other infeasibilities. The desirability of one method versus the other is, perhaps, mostly a matter of computer programming, and we will not discuss it further.

In order to remove the infeasibility of a non-negative variable s_i (assuming that $a_{i,0} < 0$) we consider again the subproblem formed with those rows that are already feasible, and having as objective the maximizing of s_i. Only those rows that are (lexicographically) negative within the subproblem will be considered for pivoting. If, at some stage, there is no such a column, and if $a_{i,0} < 0$, this means that s_i cannot reach a non-negative value in the subproblem, and therefore there is no feasible solution of the original problem. If the subproblem is unbounded in some column j (such that $a_{i,j} < 0$), we can then generate a cut from row i by letting $p = | a_{i,j} |$. The Gomory cut will have $b_0 < 0$ and $b_j < 0$ and after pivoting on b_j, $a_{i,0}$ will be non-negative (this is the type of cuts that Gomory uses in [6]).

But whether or not such unboundedness is found, the idea is to improve s_i until it becomes non-negative, and at the same time to maintain feasibility in the subproblem. The variable s_i does not have to be optimized in the subproblem; as soon as it becomes non-negative we take another infeasible variable as our new objective, if there are any left (in the process of "satisfying" a certain non-negative variable, some other variables may also be satisfied).

When every variable is feasible we take the original objective of the problem and we proceed to optimize it, preserving the feasibility.

This technique for finding a first feasible solution has been incorporated in our algorithm and it has been programmed for the IBM 7094. Its convergency depends, of course, on the convergency of our basic algorithm.

The ideas behind this approach are not exclusively for integer linear programming, but they can also be applied, with proper modifications, to regular linear programming.

CHAPTER 4

THE PROBLEM OF FINITE CONVERGENCE

4.1 Considerations about Convergence

As mentioned in Chapter 2, in applying the basic algorithm there will be major iterations (breakthroughs) and minor iterations. Every time a major iteration occurs, the solution vector (column a_0) increases lexicographically.

If we are using the maximizing version of the basic algorithm, we have the following theorem:

Theorem 4.1 If the problem is bounded in every direction, there can only be a finite number of major iterations.

Proof[1]: A sequence of major iterations will produce a sequence of columns a_0's that will be in lexico-increasing order and which will be bounded above by x^0, the lexico-largest feasible solution:

$$a_0^0 \overset{\ell}{<} a_0^1 \overset{\ell}{<} a_0^2 \overset{\ell}{<} \ldots \overset{\ell}{<} x^0$$

1 This proof follows to a certain extent a similar proof given by Gomory in [6].

Let us assume that we had an infinite sequence of this sort. The vectors are all-integer, so their components charge only by integer amounts. The first component (corresponding to the objective function) cannot increase indefinitely because it is bounded above by z_0, the first component of x^0. Therefore, the first component can only increase for a finite number of times and it then remains fixed at some value $z' \leq z_0$.

From now on the second component must be non-decreasing, and there are two alternatives: a) it increases indefinitely, or b) it increases up to a certain value, and then remains fixed. Alternative a) is excluded because we assume that the problem is bounded in every direction, that is, that the feasible range of every variable has an upper bound. If alternative b) occurs we turn our attention to the third component of α_0, which offers the same two alternatives. In every case alternative a) will be excluded, so that if every component increases up to a certain value and then remains fixed, the value of every component will be fixed after a finite number of major iterations. Thus, the assumption of an infinite sequence has been contradicted and the theorem is proved. (Q.E.I

However, if we use the minimizing version of the basic algorithm we have the following theorem, which is stronger than the previous one:

Theorem 4.2 If the problem is bounded in the direction of minimizing z, there can only be a finite number of major iterations.

Proof: The proof is similar to the previous one except that we have now a lexico-decreasing sequence of solution vectors, and that while z has a lower bound by hypothesis, the lower bound of every other variable is automatically provided, because they all have zero as a lower bound. (Q.E.D.)

(Any particular problem can, of course, be solved by either version of the basic algorithm, but in view of theorems 4.1 and 4.2 it appears that the minimizing version is preferable).

In order to guarantee finite convergence for the overall algorithm it is necessary to guarantee that a breakthrough (that is, a major iteration) can always be obtained after a finite number of minor iterations. One reason why a major iteration might require a very large or even an infinite number of iterations, is because at any stage there may be more than one generating element (i.e. more than one choice of $i0$, $j0$), and if proper rules of choice are not followed, some sort of cycling may then occur.

The problem of finding rules of choice that will assure finite convergence of the primal algorithm appears to be a difficult one. After a large number of analytical and computational efforts we have not succeeded in finding rules of choice that we can prove will guarantee finite convergence in solving a general problem. However, we report here two encouraging results: first a formal proof of finite convergenc of the algorithm when applied to two-dimensional problems; and second, the definition of a family of rules of choice (for the general problem) which has both analytical and computational appeal.

4.2 Finite Convergence of Two-Dimensional Problems

We have been able to prove finite convergence of the algorithm for two-dimensional integer LP problems. Consider a general two-dimensional integer LP problem. It can be expressed by means of a tableau with columns j, $j = 0, 1, 2$; and rows i, $i = 0, 1, \ldots, m$. These $m + 1$ rows include the objective function (row zero), and n constraints, two of which are the non-negativity constraints of the problem variables. The objective variable is unsigned and every other row variable is non-negative. We will assume that the tableau is already in primal feasible form since we are concerned here only with the basic algorithm. Let us say that the objective function is to be maximized, and let us assume that the problem is bounded in every direction.[1]

1 This does not entail any loss of generality because it can always be done by using A. CHARNES and W.W. COOPER "regularization" method. See their book Management Models and Industrial Applications of Linear Programming. 2 vols. (New York: John Wiley and Sons, Inc. 1961) pp. 424-427.

Types of Rows. Dominance

One of the characteristics of pivoting on an element of value + 1 (as it is the case with the basic algorithm) is that the pivoting column (α_{j0}) changes sign; and in particular, it becomes (lexico-graphically) positive.

Considering the two columns (α_1 , α_2), there are three possibilities:

a) Both are positive,

b) One is negative, the other one is positive,

c) Both are negative.

In case a), optimality is already proved, so the problem is solved. In either case b) or c) we can pivot on a negative column and it will become positive, while the other column either becomes (or stays) positive or negative. If positive, optimality has been proved; if negative, we are left with a negative column and a positive column.

Whenever one column is negative and the other one is positive (the zeroth column has been excluded from these considerations), we say that the tableau is in <u>normal</u> <u>form</u>. We can see that the tableau will remain in normal form until optimality is proved. (The pivoting column always becomes positive and the other one becomes negative, unless optimality is proved). We will denote by "a" the negat column (its elements will be a_{ia}) and by "b" the other column (elements a_{ib}).

A row i, $i \neq 0$, will be classified according to the signs of its elements a_{ia} and a_{ib} :

Type	a_{ia}	a_{ib}
1	0	-
2	-	0
3	-	-
4	-	+
5	0	+
6	+	0
7	+	-
8	+	+

Furthermore, let us divide the rows of type 4 into two sub-types.

4a) If $a_{0a}\, a_{ib} \geqslant a_{0b}\, a_{ia}$

4b) If $a_{0a}\, a_{ib} < a_{0b}\, a_{ia}$

Any row of type 1, 2, 3 or 4a will be called a dominated row. Any row of type 4b, 5, 6, 7 or 8 is a non-dominated row. The current level is the number of non-dominated rows in the current tableau. Rows of type 1, 2 or 3 are called trivial. Only rows of type 6, 7 or 8 are pivotable because they are the only ones with $a_{ia} > 0$.

Considerations about Pivoting

When the tableau is in normal form and a pivot is performed, columns 1 and 2 exchange roles (if optimality is not proved). That is, if column 1 was designated as column "a" before pivoting, it will become column "b" after pivoting; and vice versa. The same is true for column 2, of course.

The pivoting formulas (2.4.3) may then be expressed as:

$$a_{i0}^{k+1} = a_{i0}^{k} - b_{0}^{k}(i0)\, a_{ia}^{k}$$

$$a_{ia}^{k+1} = a_{ib}^{k} - b_{b}^{k}(i0)\, a_{ia}^{k} \qquad (4.2.1)$$

$$a_{ib}^{k+1} = -\, a_{ia}^{k} \qquad i = 0, \ldots, m$$

where $b_{j}^{k}(i) = [a_{i,j}^{k}/a_{i,a}^{k}] \qquad j = 0,\ a,\ b;\quad a_{ij}^{k} > 0 \qquad (4.2.2)$

The quantities $b_{b}^{k}(i)$ will be called <u>b-factors;</u> they are defined only for pivotable rows. (The superscript k indicates that the quantity corresponds to the k^{th} tableau).

Some Lemmas for the Tableau in Normal Form

Let ${}^{+}I_{a}^{k} = \{i \mid i = 1, \ldots, m \text{ and } a_{ia}^{k} > 0\}$.

<u>Lemma 4.1</u>: If after pivoting (on column a) on a cut generated from i0 ($i0 \in {}^{+}I_{a}^{k}$), optimality is not proved, then the following conditions hold for any row i , $i \in {}^{+}I_{a}^{k}$:

Condition A) If $b_b^k(i) > b_b^l(i0)$ then $b_b^{k+1}(i) = -1$

Condition B) If $b_b^k(i) = b_b^k(i0)$ then either $b_b^{k+1}(i) < -1$

or row i becomes trivial.

Condition C) If $b_b^k(i) < b_b^k(i0)$ then row i becomes trivial.

<u>**Proofs:**</u>

Condition A: Let $d = b_b^k(i) - b_b^k(i0)$, we will then have $d \geqslant 1$ because $b_b^k(i) > b_b^k(i0)$ and the b-factors are integer.

Applying (4.2.1)

$$a_{ia}^{k+1} = a_{ib}^{k} - b_b^k(i0)\, a_{ia}^{k} = a_{ib}^{k} - b_i^k(i)\, a_{ia}^{k} + d\, a_{ia}^{k}$$

By means of (3.2.2) and (4.2.2)

$$a_{ia}^{k+1} = (a_{ib}^{k} \mid a_{ia}^{k}) + d\, a_{ia}^{k}$$

Therefore, because of (3.2.1) and because $d \geqslant 1$,

$$a_{ia}^{k+1} \geqslant a_{ia}^{k} > 0$$

Also $a_{ib}^{k+1} = -\, a_{ia}^{k} < 0$

so that $|a_{ib}^{k+1}| \leqslant a_{ia}^{k+1}$

then $b_b^{k+1}(i) = [a_{i,b}^{k+1}/a_{i,a}^{k+1}] = [-|a_{ib}^{k+1}|/a_{ia}^{k+1}]$ (Q.E.D.)

Condition B: If $b_b^k(i) = b_b^k(i0)$ we will have by (4.2.2), (3.2.2) and (3.2.1),

$$a_{ia}^{k+1} = (a_{ib}^{k} \mid a_{ia}^{k})$$

and $a_{ia}^{k} > a_{ia}^{k+1} \geq 0$

also, $a_{ib}^{k+1} = -a_{ia}^{k} < 0$

1) if $a_{ia}^{k+1} = 0$, then row i becomes of type 1, that is, a <u>trivial</u> <u>row</u>.

2) If $a_{ia}^{k+1} > 0$, then $|a_{ib}^{k+1}| = a_{ia}^{k} > a_{ia}^{k+1}$

therefore

$$b_{b}^{k+1}(i) = [a_{ib}^{k+1}/a_{ia}^{k+1}] = [-|a_{ib}^{k+1}|/a_{ia}^{k+1}] < -1 \qquad \text{(Q.E.D.)}$$

<u>Condition C</u>: If $b_{b}^{k}(i) < b_{b}^{k}(i0)$ and we let $d = b_{k}^{k}(i0) - b_{b}^{k}(i)$

then $d \geq 1$

now

$$a_{ia}^{k+1} = a_{ib}^{k} - b_{b}^{k}(i0)\, a_{ia}^{k} = a_{ib}^{k} - b_{b}^{k}(i)\, a_{ia}^{k} - d\, a_{ia}^{k} = (a_{ib}^{k} \mid a_{ia}^{k}) - d\, a_{ia}^{k}$$

but

$$(a_{ib}^{k} \mid a_{ia}^{k}) < a_{ia}^{k} \quad \text{and} \quad d\, a_{ia}^{k} \geq a_{ia}^{k}$$

so that

$$a_{ia}^{k+i} < 0$$

also,

$$a_{ib}^{k+1} = -a_{ia}^{k} < 0$$

then, row i becomes of type 3, that is, a <u>trivial</u> <u>row</u>. (Q.E.D.)

Lemma 4.2: Every pivot, except perhaps the last, is done on a cut generated from a row of type 7.

Proof: In normal form the only pivotable rows are those of type 6, 7 or 8.

If we pivot on a cut generated from either a type 6 or a type 8, we will have $b_b^k(i0) \geq 0$ and

$$a_{0a}^{k+1} = a_{0b}^k - b_b^k(i0)\, a_{0a}^k > 0$$

(because by assumption, $a_{0a}^k < 0$ and $a_{0b}^k \geq 0$ for $k \geq 1$)

also, $a_{0b}^{k+1} = - a_{0a}^k > 0$.

The solution would have been proved optimal and the algorithm would stop. Therefore, only the last cut upon which we pivot may come from a row of type 6 or 8; all the other cuts must come from rows of type 7. (Notice that optimality can also be proved after pivoting on a cut generated from a type 7 row).

Lemma 4.3: Every dominated row stays dominated, unless optimality is proved.

Proof: A dominated row may be of type 1, 2, 3 or 4a. If row i is of type 1, after pivoting (on a cut generated from a type 7 row), we will then have

$$a_{ia}^{k+1} = a_{ib}^k - b_b^k(i0)\, a_{ia}^k = a_{ib}^k < 0 \; ; \; a_{ib}^{k+1} = - a_{ia}^k = 0$$

thus, it becomes a row of type 2, and therefore, it remains dominated.

For rows of type 2, 3 and 4a, we may restate lemma 4.3 in the following fashion:

<u>Lemma 4.3'</u>: Any row with $a_{ia}^{k} < 0$ and a_{ib}^{k} such that $a_{0a}^{k} a_{ib}^{k} \geq a_{0b}^{k} a_{ia}^{k}$ will become, in one iteration and if optimality is not proved, a row of type 4a.

This lemma states that rows of types 2 and 3 become of type 4a; and those of type 4a remain of type 4a. This agrees wi h lemma 4.3.

<u>Proof of lemma 4.3'</u>: If optimality is not shown we shoulc have $a_{0a}^{k+1} < 0$, because we already know that $a_{0b}^{k+1} = - a_{0a}^{k} > 0$. Then,

$$a_{0a}^{k+1} = a_{0b}^{k} - a_{0a}^{k} b_{b}^{k}(i0) < 0$$

or

$$(a_{0b}^{k}/a_{0a}^{k}) > b_{b}^{k}(i0) \qquad (4.2.3)$$

We want to prove that row i, after pivoting, will be a row of type 4a, that is, such that

$$a_{0a}^{k+1} a_{ib}^{k+1} \geq a_{0b}^{k+1} a_{ia}^{k+1}, \text{ with } a_{ia}^{k+1} < 0, \; a_{ib}^{k+1} > 0 \quad .$$

Applying (4.2.1)

$$a_{0a}^{k+1} = a_{Cb}^{k} - b_{b}^{k}(i0) a_{0a}^{k} < 0 \qquad \text{(by hypothesis)}$$

$$a_{0b}^{k+1} = - a_{0a}^{k} > 0$$

$$a_{ia}^{k+1} = a_{ib}^{k} - a_{ia}^{k} b_b^k (i0)$$

$$a_{ib}^{k+1} = - a_{ia}^{k} > 0 \qquad (4.2.4)$$

From the original hypothesis and from (4.2.3)

$$(a_{ib}^k / a_{ia}^k) \geq (a_{0b}^k / a_{0a}^k) > b_b^k (i0)$$

remembering that $a_{ia}^k < 0$, we have $a_{ib}^k < b_b^k (i0)\, a_{ia}^k$

and therefore $a_{ia}^{k+1} < 0$.

Let us now assume that $a_{0a}^{k+1} a_{ib}^{k+1} < a_{0b}^{k+1} a_{ia}^{k+1}$.

If we substitute the expressions (4.2.4) into this inequality, after simplifying we obtain

$$a_{0a}^k a_{ib}^k < a_{0b}^k a_{ia}^k$$

which contradicts the original hypothesis. Therefore we should have:

$$a_{0a}^{k+1} a_{ib}^{k+1} \geq a_{0b}^{k+1} a_{ia}^{k+1}$$

We have also proved that $a_{ia}^{k+1} < 0$ and $a_{ib}^{k+1} > 0$; so that the proof of lemma 4.3' (and of lemma 4.3) is now complete.

Lemma 4.4: A row of type 4b, in a single iteration, and if optimality is not proved, will either remain of type 4b, or become or type 5 or 8, but it cannot become of type 4a.

Proof: By definition, a row of type 4b has $a_{ia}^k < 0$, $a_{ib}^k > 0$ and $a_{0a}^k a_{ib}^k < a_{0b}^k a_{ia}^k$.

We will have

$$a_{ia}^{k+1} = a_{ib}^{k} - b_{b}^{k}(i0)\, a_{ia}^{k}$$

$$a_{ib}^{k+1} = -\, a_{ia}^{k} > 0 \quad .$$

We cannot say anything certain about the sign of a_{ia}^{k+1} ; it may be positive, zero, or negative. If it is positive, row i has become of type 8; if it is zero, row i has become of type 5; if it is negative, row i has remained of type 4.

Let us assume that $a_{ia}^{k+1} < 0$, and that row i has become of type 4a, that is, that $a_{0a}^{k+1}\, a_{ib}^{k+1} \geqslant a_{0b}^{k+1}\, a_{ia}^{k+1}$. Putting this in terms of elements of the k^th tableau and simplifying we get

$$a_{0a}^{k}\, a_{ib}^{k} \geqslant a_{0b}^{k}\, a_{ia}^{k}$$

but this contradicts the original hypothesis; so that we should have $a_{0a}^{k+1}\, a_{ib}^{k+1} < a_{0b}^{k+1}\, a_{ia}^{k+1}$, and this proves that row i may remain of type 4b, but cannot become of type 4a.

<u>Lemma 4.5</u>: A row of type 5 will become, in a single iteration and if optimality is not shown, a row of type 6.

<u>Proof</u>: If row i is of type 5, then by definition

$$a_{ia}^{k} = 0 \ , \ a_{ib}^{k} > 0$$

after pivoting:

$$a_{ia}^{k+1} = a_{ib}^{k} - b_{b}^{k}(i0)\, a_{ia}^{k} = a_{ib}^{k} > 0$$

$$a_{ib}^{k+1} = - a_{ia}^{k} = 0$$

Thus, row i has become of type 6.

Lemma 4.6: A row i of type 6 or 8 will become, in a single iteration and if optimality is not proved, a row of type 7. Furthermore, it will have $b_{b}^{k+1}(i) = -1$.

Proof: By definition, if row i is of type 6 or 8, it will have $a_{ia}^{k} > 0$, $a_{ib}^{k} \geq 0$.

If optimality is not proved this means that we pivoted on a cut generated from a row of type 7 (lemma 4.2). Therefore, we will have $b_{b}^{k}(i0) \leq -1$.

Then

$$a_{ia}^{k+1} = a_{ib}^{k} - b_{b}^{k}(i0)\, a_{ia}^{k} \geq a_{ia}^{k} > 0$$

$$a_{ib}^{k+1} = - a_{ia}^{k} < 0 \quad .$$

Thus, row i is now of type 7.

Also,

$$b_{b}^{k+1}(i) = [a_{ib}^{k+1}/a_{ia}^{k+1}] = -1 \quad \text{because} \quad -1 \leq (a_{ib}^{k+1}/a_{ia}^{k+1}) < 0 \quad .$$

Lemma 4.7: A row i of type 7 may become, in a single iteration and if optimality is not proved, a row of type 1 or 3, or it may remain of type 7.

Proof: By definition, if row i is of type 7 it will have $a_{ia}^{k} > 0$, $a_{ib}^{k} < 0$. Then,

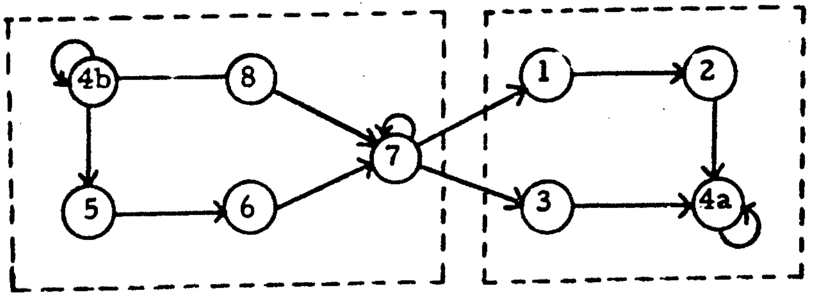
$$a_{ia}^{k+1} = a_{ib}^{k} - b_{b}^{k}(i0)\, a_{ia}^{k}$$

$$a_{ib}^{k+1} = -\, a_{ia}^{k} < 0$$

If optimality is not proved this means that $b_{b}^{k}(i0) \leq -1$. Now, a_{ib}^{k+1} is necessarily negative, but a_{ia}^{k+1} is the difference of two negative quantities; therefore, it may either be positive, negative, or zero. Row i will then become of type 7, 3 or 1 resepctively.

The Graph of Changes of Type

From lemmas 4.3 to 4.7 we realize that a row may change its type in a single iteration (and if optimality is not shown) but only to certain other types. The possible changes can be summarized with the following graph:



Non-dominated rows Dominated rows

It will be noticed that there is a possible flow of non-dominated rows, but not vice versa; that is, we have proved that the level is non-increasing.

A Theorem on Convergence of Two-Dimensional Problems

Lemma 4.8: Unless optimality is proved, or unless a breakthrough is found, the level can be decreased by at least one in a finite number of iterations.

Proof: If there were no pivotable rows in the k^{th} tableau, the problem would have an unbounded solution and this would contradict our initial assumption.

If there are only pivotable rows of type 6 or 8, but none of type 7, then at least one of them is most-binding and the solution will be proved optimal as was shown in lemma 4.2.

But let us assume, for the sake of argument, that there is at least one row of type 7, and that no row of type 6 or 8 does ever become most-binding. Of the set of rows of type 7 there will be some rows i with $b_b^k(i) = -1$ and some with $b_b^k(i) < -1$. Let us call the two subsets ${}^7I_1^k$ and ${}^7I_2^k$, respectively. There will be two cases:

Case I: ${}^7I_2^k$ empty (and ${}^7I_1^k$ not empty)

Case II: ${}^7I_2^k$ not empty (${}^7I_1^k$ either empty of not empty)

In case I, after one iteration and if optimality is not proved, we will have case II; because by condition B, lemma 4.1, every row in ${}^7I_1^k$ will be in ${}^7I_2^{k+1}$. (In performing this iteration, one or more of the rows in ${}^7I_1^k$ may become trivial, in which case the lemma is also satisfied.)

In case II, if there is any most-binding row in ${}^7I_1^k$ and we pivot on the cut generated from it, then every row of ${}^7I_2^k$ will become trivial, because $b_b^k(i0) = -1$ and because of condition C of lemma 4.1; and the level would therefore decrease, thus satisfying the lemma.

So, let us assume that at least one row in ${}^7I_2^k$ is most-binding and that we pivot on the cut generated from it (and optimality is not proved upon pivoting). Every row in ${}^7I_1^k$ will be in ${}^7I_1^{k+1}$ as condition A of lemma 4.1 shows (because $b_b^k(i0) < -1$ and $b_b^k(i) = -1$ for every $i \epsilon {}^7I_1^k$). Any rows that were of type 6 or 8 will come to belong to ${}^7I_1^{k+1}$ as lemma 4.6 shows. Furthermore, any row of ${}^7I_2^k$ which had $b_b^k(i) > b_b^k(i0)$ will become of the set ${}^7I_1^{k+1}$ because of condition A of lemma 4.1. If there were any rows in ${}^7I_2^k$ such that $b_b^k(i) < b_b^k(i0)$, then, by condition C of lemma 4.1, they would become trivial and the lemma would be satisfied.

The only rows that would belong to ${}^7I_2^{k+1}$ would have belonged to ${}^7I_2^k$ and would have had $b_b^k(i) = b_b^k(i0)$. The only way the level could remain unchanged is if we could always be able to find a most-binding row in the sets ${}^7I_2^h$, $h = k, k+1, \ldots,$ and if none of the rows in these sets became trivial.

However, any row i in ${}^7I_2^h$ would have had $b_b^k(i) = b_b^k(i0)$, $b_b^{k+1}(i) = b_b^{k+1}(i0), \ldots, b_b^h(i) = b_b^h(i0)$, and therefore, because of lemma 3.2 and (4.2.1) we will have

$$a_{ia}^k > a_{ia}^{k+1} > \ldots > a_{ia}^{h-1} > a_{ia}^h \geq 0 \quad .$$

Clearly, because the a_{ia}'s are integer and a_{ia}^k is finite, there will be a tableau h in which every row i, $i \in {}^7I_2^h$, would be such that

$$a_{i0}^h > a_{ia}^h \geq 0$$

This means that: either at least one of the rows originally in ${}^7I_2^k$ became trivial at some tableau h (when $a_{ia}^h = 0$); and/or a tableau h has been found in which no row of ${}^7I_2^h$ is zero-binding. Therefore, if there is any zero-binding row in ${}^7I_1^h$ we are forced to pivot on the cut generated from it and thus every row in ${}^7I_2^h$ would become trivial and the lemma would be proved. (The set ${}^7I_2^h$ would not be empty because by condition B of lemma 4.1, it would contain at least the row from which the last cut was generated; unless that row has become trivial, in which case the lemma is also satisfied.) Otherwise, if there is no zero-binding row in ${}^7I_1^h$ either, this means that a breakthrough has been found, and the lemma is again satisfied. (Q.E.D.)

<u>Theorem 4.3:</u> If the two-dimensional problem is bounded (as in lemma 4.8) then, in a finite number of minor iterations (pivots), an optimal solution will be found and its optimality will be proved.

Proof: Any tableau in normal form has a finite number of non-dominated rows (i.e. the level is finite); therefore, by repetitive application of lemma 4.8 we see that, unless optimality is proved or a breakthrough is found, the level will decrease, and it could become zero in a finite number of iterations. But this would imply that the problem is unbounded and this is contrary to our assumptions. Therefore, in a finite number of iterations either a breakthrough is found or optimality is proved. This, together with theorem 4.1 proves the present theorem.

(Q.E.D.)

It should be pointed out that neither finding a breakthrough, nor pivoting after it has been found, does necessarily imply a decrease in the level.

Comments on the Proof of Finite Convergence

It should be noticed that the proof, besides boundedness of the problem, only requires that the tableau be in normal form; and this can be accomplished by using at most one iteration; but it does not require the use of any specific rule for choosing one among the several most-binding rows that might be available for pivoting, and it does not require that rows of type 6 or 8 (which imply immediate optimality) be recognized.

However, the performance of the algorithm with a two-dimensional problem can be improved greatly if the following rule is used.

Rule: Among the available most-binding rows (in case there are ties) choose i0 such that

$$b_b^k(i0) = \max_{i \in I} \{b_b^k(i)\}$$

where I is the set of most-binding rows.

This rule has two convenient effects: a) it accelerates the decreasi of the level, b) it "recognizes" and gives preference to rows of type 6 or 8, thus shortening the process.

4.3 A Promising Family of Rules of Choice for n-Dimensional Convergen

Among the many rules of choice that were tried analytically and computationally for convergence of general problems, one class of rules finally emerged as the most reasonable, and it has indeed given encouraging computational results. We have called this family of rules: hierarchical rules of choice. Let us consider the n current k-dimension problems P(k), constituted by the first k current parametric variables $(1 \leq k \leq n)$ plus the zero th column. This establishes a current hierarchy of problems; that is, P(k) is a sub-problem of P(k + 1), which in turn is a sub-problem of P(k + 2), and so forth. P(n) is, of course, the original problem.

The rules will be presented by giving the algorithm.

Hierarchical Algorithm

A) If there is no breakthrough on any column go to C, otherwise go to B.

B) From the columns with a breakthrough, select the most negative one as the pivoting column. Choose one of the most-binding rows as generating column, generate the Gomory cut, pivot and go back to A.

C) Select the smallest not lexico-optimal problem P(k). That is, k is chosen so that $a_k^{\ell} < 0$ while $a_j^{\ell} > 0$, j = 1, . . . , k - j . (This implies that the current P(k-1) has been proved lexico-optimal.) If there are no negative columns the lexico-optimal solution has been found and its lexico-optimality has been proved.

D) In order to solve the current P(k) we extract from it a (k - 1) dimensional not lexico-optimal sub-problem. This means that we have to <u>reorder</u>, in some specific way, the first k columns so that the negative column a_k will now occupy one of the first k - 1 places (that is, without counting column zero). This new first k - 1 column will define our <u>new</u> P(k - 1). This reordering is done recursively so that a new hierarchy of sub-problems P(j) , j = 1, . . . , k - 1 is established. The negative column (previously the kth) will now occupy the first position and, together with a_0 , constitutes the new P(1). The first column is now chosen as the pivoting one.

E) Considering again the complete tableau, one of the rows that is most-binding on column 1 is selected as the generating row, a cut is generated from it, and after pivoting (thus possibly modifying the whole tableau) one goes back to A.

In step D we implicitly extract from $P(k)$, by reordering, a new $P(k - 1)$ because the current one has been proved lexico-optimal. It is in this re-extracting of a sub-problem that the main theoretical difficulty lies, because in order to assure convergence, this new $(k - 1)$-dimensional sub-problem has to "dominate", in some fashion, the current $P(k - 1)$ whose lexico-optimality has just been established. In other words, if the current $P(k - 1)$ has been proved lexico-optimal, it means that no better solution for $P(n)$ is found in the $(k - 1)$-dimensional polyhedron defined by $t_j = 0$, $j = k, \ldots, n$. Thus, we have to change one of those parametric variables specifically, by reordering the first k columns as described above, we bring in a new t_k. This new t_k would have to be more "dominating" than the old one. Although this dominance has been established for 2-dimensional problems, it has not been generalized to larger problems. The way we have used to go around this difficulty is described next.

Some Heuristic Members of the Family

Since no analytical grounds for extracting the sub-problems have been discovered, we have resorted to heuristics (and, anyway, heuristics are usually the roads to analytical grounds).

If the current P(k - 1) is lexico-optimal and $a_k < 0$, extracting a new P(k - 1) from P(k) is equivalent to upgrading one of the first (k - 1) columns to the k^{th} place and making a_k one of the first (k - 1) columns. It is in selecting that column of P(k - 1) which is to be upgraded, that a heuristic can be introduced. Some possible heuristics are: a) upgrade the $(k - 1)^{th}$ column (heuristic JA), b) upgrade the first column (heuristic JB), c) upgrade the column upon which we have pivoted most recently (heuristic JC), d) upgrade the lexico-largest column (heuristic JD).

Rule A when applied recursively, would put the k^{th} column in the first place and would shift the first (k - 1) columns one place up. Rule B would reverse the order of the first k columns. Rule D would rearrange the first k columns in lexico-decreasing order. In any case, the pivoting column (which previously was the k^{th}) would end up in the first position.

Once the pivoting column has been chosen, and the columns have been reordered it is necessary to choose a pivoting row among the ones that are most-binding on the pivoting column. The following are two of the possible heuristic rules of choice: a) take the first most-binding row (heuristic IA), b) take the most-binding row that generates the lexico-largest Gomory cut (heuristic IB). Heuristic IB is a direct extension of the rule given at the end of section 4.2. (A row vector B_i is lexicographically larger than another row vector B_j if the first element where they differ, counting from left to right, is larger in B_i than in B_j.)

4.4 Computational Experience

We have not tried out every combination of heuristics nor even every heuristic. Only the following three combinations have been studied JA - IA, JA - IB, and JB - IB. Combinations JA - IA and JA - IB were successful with many problems but JA - IB was more efficient: JA - IB solved problems that were not solved (in a reasonable amount of time) by JA - IB, and those problems that were solved by both combinations were solved faster by JA - IB.

A few small problems were tried with combination JB - IB but none were solved. Thus, we have selected combination JA - IB as a good heuristic.

In order to compare the performance of our algorithm with those of other authors we have selected the nine examples found in ref. [10]. Thompson has used those examples to compare his method with the all-integer method of Gomory [6]. The results are shown in Table 4.1.

In analysing these results it should be considered that the iterations of Gomory's all-integer algorithm are computationally equivalent to ours. Thompson reports two figures for performance: the number of stopped linear programming problems that were solved (L. P.'s), and the total number of pivots required for their solution (last column of Table 4.1). He also states that the solution of one of the stopped LP problems takes about the same as one of Gomory's iterations, whether it is done by hand or by machine.

The nine problems are initially in dual feasible form which would seem to be advantageous to both Gomory's and Thompson's algorithms.

Our algorithm was programmed in such a way that whenever several columns had a breakthrough, the first of these columns was chosen, rather than the most negative.

TABLE 4.1 NINE EXAMPLES SOLVED WITH DIFFERENT METHODS

Example	Dimensions of Tableau	Rule JA - IB			Optimal Found		
		Feasible Found	Optimal Found	Optimal Proved[1]	GOMORY (iterations)	THOMPSC	
						L. P.'s	Pi
		(Iteration No.)					
1	5 x 3	5	7	8	13	20	
2	7 x 4	35	37	37	1000*	?	
3	7 x 4	24	25	84	1570	173	1
4	11 x 6	500*	-	-	?	?	
5	5 x 3	51	102	102	?	255	1
6	5 x 3	102	102	202	?	260	1
7	5 x 3	102	102	202	?	255	1
8	17 x 9	4	12	74	5215	13	
9	18 x 8	1000*	-	-	?	2038	108

Other Problems Tried

A	8 x 6	0	17	17
B	20 x 13	14	19	20
C	25 x 13	78	-	300*
D	43 x 19	16	22	38

The asterisk (*) indicates that the program was stopped after this many iterations without having found the answer.

(1)This is the total number of iterations required to solve the problem (it includes the iterations used for finding a first feasible solution). A question mark (?) indicates that the figure was not reported in ref. [10]. It does not imply that the problem was not solved.

CHAPTER 5

INTEGER L. P. AND NUMBER THEORY

In the present chapter we will develop a generalization of the Euclidean Algorithm, and we will explore the application of integer L. P. to the solution of linear diophantine equations. The importance of these problems and of their solution is well established in number theory (for instance, see refs. [23] and [24]).

5.1 Generalized Euclidean Algorithms

Whenever we talk of the greatest common divisor (g. c. d.) of two or more numbers, or of a vector of numbers, it will be with the implicit understanding that the numbers are integer, and that we are referring to the g. c. d. of their absolute value.

The Euclidean Algorithm finds the g. c. d. of two numbers by performing successive divisions (where the divisor is the previous remainder and the dividend is the previous divisor), until the remainder of a division becomes zero. The absolute value of the last divisor, which is also the last non-zero remainder, is the g. c. d. of the two original numbers. The procedure terminates in a finite number of steps.

The traditional way of finding the g. c. d. of more than two numbers has been to find first, by means of the Euclidean Algorithm, the g. c. d. of two of the numbers; say it is d_1. Then the g. c. d. of d_1 and another one of the numbers is found, and so forth, until all of the numbers have been taken into consideration. The final number obtained is shown to be the g. c. d. of all the original numbers taken together.

We propose here a more general procedure, which is based in what we call elementary operations on a vector. Consider a vector $A = (a_1, \dots, a_n)$. An elementary operation is either: changing the sign of one element, or, adding or subtracting from element a_i an integer multiple of another element a_j, $j \neq i$. Thus, a_i will be called the element to be transformed, and a_j the transforming element.

Lemma 5.1: The g. c. d. of a vector is unchanged by elementary operations.

Proof: Let $A = (a_1, a_2, \dots, a_n)$, where the a_i's are integer and at least one of them is different from zero. Let d be the g. c. d. of A. (If there are any zero elements this does not make any difference because any number is the divisor of zero. Thus, the g. c. d. of vector A as a whole is the same as the g. c. d. of the non-zero elements of A).

One of the two types of elementary operations is to change the sign of one of the elements; this will obviously not affect the g. c. d. of the vector.

The other type of elementary operation is where a_i is changed into $a_i + k\,a_j$; $i \neq j$, $k = 0, \pm 1, \pm 2 \ldots$. Let us call A' the vector A after we have applied one elementary operation of the last type. We want to show that the g. c. d. of A' is the g. c. d. of A. Say that d is the g. c. d. of A. We have to show that d is a common divisor of A', and that there is no larger common divisor of A'. If d is the g. c. d. of A we can put

$$A = (a_1, \ldots, a_n) = (db_1, db_2, \ldots db_n)$$

After the elementary operation the only element that will be changed will be a_i'. We will have

$$a_i' = a_i + ka_j = db_i + kdb_j = d(b_i + kb_j)$$

Thus, d will be a common divisor of A'. Let A_i be the $(n - 1)$-element vector that remains after we remove a_i from A. The g. c. d. of A_i is either d or a multiple of d. If it is d, then the g. c. d. of A', $A' = (A_i, a_i') = (A_i, d(b_i + kb_j))$, is also d because of the associative law of the g. c. d. (see ref. [23], p. 48).

On the other hand, let us assume that the g. c. d. of A_i is cd, where c is a positive integer larger than one. The only way in which the g. c. d. of A' would be larger than d, would be if the g. c. d. of cd and $a_i' = d(b_i + kb_j)$, was larger than d ; and this could only occur if c and b_i had some common divisor, as d_1, $(d_1 > 1)$. But if this were true, then the g.c.d. of A would have been d_1 in the first place. It follows that c and b_i do no have any common divisor, and therefore, d is the g.c.d. of A' , which proves the lemma. (Q.E.D.)

We can now propose the following theorem that justifies the Generalized Euclidean Algorithm (G.E.A.).

Theorem 5.1: If we have an n-dimensional vector and we perform a sequence of elementary operations on it, so that only one non-zero element remains; then the absolute value of this last number is the g.c.d. of the original vector.

Proof: The g.c.d. of a vector that has only one non-zero element is precisely the absolute value of that element. Therefore, by lemma 5.1 it follows that this is the g.c.d. of the original vector. (Q.E.D.)

Any rule that guarantees the "reduction" of the original vector (so that it will have only one non-zero element) in a finite number of steps, will constitute a valid G.E.A. The following rule provides a family of convergent G.E.A.'s.

Rule: Always apply elementary operations such that the absolute value of the transformed element is smaller than its previous absolute value, and smaller than the absolute value of the transforming element.

The Euclidean Algorithm is a particular case of the G.E.A. because finding the remainder of the division is an elementary operation that satisfies the previous rule.

These results, although they are somewhat stronger and have been developed independently, are similar to those of W. A. Blankinship (ref. [22]). He not only finds the g. c. d. of n positive integers, but also finds one solution to the diophantine linear equation that has these n integers as coefficients and the corresponding g. c. d. as constant term. His approach does not explicitly involve integer L. P.

In the following sections we will develop, by means of integer L. P., much more general results for the solution of systems of diophantine equations.

5.2 General Solution of Linear Diophantine Equations

A linear diophantine equation is a linear equation with two or more unknowns, whose coefficients are assumed integral. The problem is to find a set of integers, or a family of sets, that satisfy the equation.

The classical way of solving an equation with two variables is by applying certain recursive relationships to the quotients and remainders obtained from the Euclidean Algorithm, when applied to the coefficients of the two variables. This results in two expressions for the unknowns, in terms of one arbitrary integer variable. An equation with n unknowns ($m > 2$) is solved by solving a sequence of equations in two variables (method of repeated reductions). The final general solution involves $n - 1$ arbitrary integer variables. (See ref. [24], Chapter III).

A system of m linear diophantine equations with n unknowns, $(m < n)$, is solved by obtaining the general solution of one of them (in terms of $n - 1$ arbitrary integer variables) and by introducing the resulting expressions into a second equation; the general solution of this second equation will now involve $n - 2$ arbitrary variables. The process is repeated until every equation is considered. The final solution for the n unknowns will involve $n - m$ arbitrary integer variables.

General Solution of Systems of Equations

We propose here a more direct method of solution of systems of linear diophantine equations. We will consider a tableau with $m + n$ rows, where the first m rows correspond to the m equations and will have associated with them m zero-variables $s_1, \ldots, s_m$. The last m rows will correspond to the n unknowns $x_1, \ldots, x_n$, and these will be considered unsigned variables (in the next section we will study the non-negative solution, here we want the general solution).

The method is to apply, to the first row, the row reduction procedure of lemma 3. 3, until there is only one non-zero element besides, perhaps, the zero^th element $(a_{1,0})$. Let the non-zero element be $a_{1,j}$, $j \neq 0$. Then we know from lemma 3.2 that $0 \leq a_{1,0} < |a_{1,j}|$. If

$a_{1,0} > 0$, then equation 1 does not have any solution in integers. (We would have $s_1 = a_{1,0} - a_{1,j}\, t_j$, but there is no integer value of t_j that makes s_1 equal to zero because $a_{1,0}$ is not a multiple of $a_{1,j}$). However, if $a_{1,0} = 0$, this means that t_j is a zero-variable and column j may be erased, thus reducing the size of the tableau. The procedure is repeated with the next row (now with only $n - 1$ variables), and the next, until every one of the m rows has been reduced, and we are left with n rows representing the general solution of the original m equations. The procedure is guaranteed to end in a finite number of steps (see lemma 3.3). For an alternate procedure see Appendix B.

Example

Find the general solution of the diophantine system:

$$3x_1 - 6x_2 + 16x_3 = 1$$

$$2x_1 + 5x_2 - 6x_3 = 2$$

The steps in the solution are:

$$\begin{array}{c|cccc|}
 & 1 & -t_1 & -t_2 & -t_3 \\
\hline
s_1 & 1 & 3^{\circ} & -6 & 16 \\
s_2 & 2 & 2 & 5 & -6 \\
x_1 & 0 & -1 & 0 & 0 \\
x_2 & 0 & 0 & -1 & 0 \\
x_3 & 0 & 0 & 0 & -1 \\
\hline
t_4) & 0 & 1* & -2 & 5
\end{array}
\qquad
\begin{array}{c|cccc|}
 & 1 & -t_4 & -t_2 & -t_3 \\
\hline
s_1 & 1 & -3 & 0 & 1^{\circ} \\
s_2 & 2 & -2 & 9 & -16 \\
x_1 & 0 & 1 & -2 & 5 \\
x_2 & 0 & 0 & -1 & 0 \\
x_3 & 0 & 0 & 0 & -1 \\
\hline
t_5) & 1 & -3 & 0 & 1*
\end{array}
\qquad
\begin{array}{c|ccc:c|}
 & 1 & -t_4 & -t_2 & -t_5 \\
\hline
s_1 & 0 & 0 & 0 & -1 \\
\hdashline
s_2 & 18 & -50 & 9^{\circ} & 16 \\
x_1 & -5 & 16 & -2 & -5 \\
x_2 & 0 & 0 & -1 & 0 \\
x_3 & 1 & -3 & 0 & 1 \\
\hline
t_6) & 2 & -6 & 1* & -
\end{array}$$

$\leftarrow\leftarrow$ t_5 is a zero-variable, the tableau may be reduced

$$
\begin{array}{c|ccc|}
 & 1 & -t_4 & -t_6 \\
\hline
s_2 & 0 & 4^{\circ} & -9 \\
x_1 & -1 & 4 & 2 \\
x_2 & 2 & -6 & 1 \\
x_3 & 1 & -3 & 0 \\
\hline
t_7) & 0 & 1* & -3
\end{array}
\quad
\begin{array}{c|ccc|}
 & 1 & -t_7 & -t_6 \\
\hline
s_2 & 0 & -4 & 3^{\circ} \\
x_1 & -1 & -4 & 14 \\
x_2 & 2 & 6 & -17 \\
x_3 & 1 & 3 & -9 \\
\hline
t_8) & 0 & -2 & 1*
\end{array}
\quad
\begin{array}{c|ccc|}
 & 1 & -t_7 & -t_8 \\
\hline
s_2 & 0 & 2^{\circ} & -3 \\
x_1 & -1 & 24 & -14 \\
x_2 & 2 & -28 & 17 \\
x_3 & 1 & -15 & 9 \\
\hline
t_9) & 0 & 1* & -2
\end{array}
\quad
\begin{array}{c|ccc|}
 & 1 & -t_9 & -t_8 \\
\hline
s_2 & 0 & -2 & 1^{\circ} \\
x_1 & -1 & -24 & 34 \\
x_2 & 2 & 28 & -39 \\
x_3 & 1 & 15 & -21 \\
\hline
t_{10}) & 0 & -2 & 1*
\end{array}
$$

$$
\begin{array}{c|cc|c|}
 & 1 & -t_9 & -t_{10} \\
\hline
s_1 & 0 & 0 & -1 \\
\hline
x_1 & -1 & 44 & -34 \\
x_2 & 2 & -50 & 39 \\
x_3 & 1 & -27 & 21 \\
\hline
\end{array}
$$

t_{10} is another zero-variable. The general solution is

$$
\begin{aligned}
x_1 &= -1 - 44\,k \\
x_2 &= 2 + 50\,k \qquad \text{where } k \text{ is an} \\
x_3 &= 1 + 27\,k \qquad \text{arbitrary integer}
\end{aligned}
$$

Analysis of Congruences

Another important part of number theory is the solution of algebraic congruences. A linear congruence can be expressed as a linear diophantine equation. For instance $a\,x \equiv b \pmod{c}$ is the same as the diophantine equation: $a\,x + c\,y = b$. Both the problem variable x , and the "modular" variable y , are unsigned.

A system of linear congruences can be solved by the method given above for the solution of systems of linear diophantine equations. One special feature is that the value of the modular variables is of no interest, therefore, we only need rows corresponding to the congruences and to the problem variables.

5.3 Non-Negative Solutions of Diophantine Equations

The problem of finding non-negative solutions (i.e. solutions in non-negative integers) of systems of linear diophantine equations is also studied in number theory. The problem may be approached by means of integer LP as will be shown in the following paragraphs.

As in the previous section, we will consider a tableau with $m + n$ rows. The first m rows correspond to the m equations and have m zero-variables associated with them. The last n rows correspond to the n unknowns, and these will now be constrained to be non-negative. The first row is taken at the same time as objective function and as a non-negative constraint; the idea is to minimize a_{10} until it becomes zero (if this is possible). But in doing this we pivot in such a way that the variables remain non-negative. Once a_{10} has become zero we may use the row reduction procedure of lemma 3.2, until there is only one non-zero element a_{1j}, $j \neq 0$ in row 1. The parametric variable associated with column j should be a zero-variable so we can now reduce the tableau. We repeat m times this procedure of reducing rows and reducing the tableau.

The result will be a tableau with n rows and $n - m + 1$ columns ($n - m$ parametric variables plus the zeroth column). The zeroth column will give one non-negative solution of the original system of equations. If we are interested in finding the lexico-smallest non-negative solution, we may apply our basic algorithm in order to minimize the solution of the reduced tableau (taking into account that the n rows correspond to non-negative variables). When every column becomes lexico-negative we would have obtained the lexico-smallest non-negative solution of the original system of equations.

The finite convergence of this procedure is, of course, dependent on the finite convergence of the basic algorithm.

If one is interested in a positive solution, it is sufficient to state the problem in terms of new variables x_i', where $x_i' = x_i - 1$, and require that the x_i' be non-negative.

CHAPTER 6

GEOMETRIC CONSIDERATIONS

6.1 The Restricted Polyhedron and the Primal Method

Any LP problem is given by a set of linear constraints and a linear objective function to be optimized. The set of constraints defines the original convex polyhedron, and it can be shown that one of the vertices of this polyhedron corresponds to an optimal solution, if an optimal solution exists. For the continuous (non-discrete) LP problem, any point on the polyhedron or in its interior will be a feasible point. The primal simplex method, through successive pivoting, advances from vertex to vertex until an optimal vertex is found; and it does this in such a way that the value of the objective function is non-decreasing as the algorithm moves from one vertex to the next.

In the case of a discrete LP problem (see section 1.1), we also have the original polyhedron, but it is no longer true that any point on its surface is a feasible point. However, there exists one, and only one, restricted convex polyhedron that is contained in the original polyhedron and such that: a) every vertex of the restricted polyhedron is a feasible point (i.e. it satisfies both the constraints and the discreteness requirements), and b) there are no feasible points outside the restricted polyhedron. (Notice that this does not imply that every point on or within the restricted polyhedron is a feasible point.) The idea of the prim algorithm is to advance (in a simplex-like manner) from a vertex of this restricted polyhedron to another vertex of the same polyhedron until the optimum is reached. By using the lexicographical simplex method the algorithm always advances to another vertex that has a larger lexicographical value (or smaller, if we are minimizing).

In order to constrain our computational "walk" to the restricted polyhedron, we have to introduce some additional inequalities or "cuts". These cuts serve only as guiding walls and as stopping barriers to constrain our walk, but they need not actually correspond to any of the planes that would define the restricted polyhedron. In the following sections we will develop these ideas further.

6.2 The Parametric Variables as a System of Coordinates

The following discussion has to do with our primal algorithm for integer LP.

The original constraints of the problem are defined in terms of $x_1, \ldots, x_n$, and are hence defined in an n-dimensional real space R^n having these variables as coordinates. But by means of linear transformations we can express the constraints in terms of different sets of coordinates. This is precisely what we accomplish by pivoting: we express all the constraints in terms of different sets of parametric variables $t_1^k, \ldots, t_n^k$. It is possible to express the constraints in terms of n arbitrary variables as long as the system of coordinates they define covers the complete space R^n. However, by obtaining the parametric variables as the result of generating Gomory-cuts (remember that the parametric variables are the slack variables of the Gomory inequalities) we achieve the following results:

a) The n coordinate planes, corresponding to the parametric variables, intersect at one of the vertices of the restricted polyhedron. This vertex is precisely the one represented by the current solution (i.e. the point defined by the current values of $x_1, \ldots, x_n$).

b) The restricted polyhedron is entirely contained in the non-negative orthant of the space defined by the parametric variables, as can be assured by the fact that these variables can only take non-negative values whenever we have a feasible (integer) solution. This characteristic is specially important for proving optimality. Notice also that this characteristic is specially important for proving optimality. Notice also that this characteristic does not imply that the original polyhedron itself will be entirely contained in the non-negative orthant. This is not the case, for instance, if the current (integer) solution is not also a vertex of the original polyhedron.

c) There is a one-to-one mapping of the lattice of integer points defined by $x_1 \ldots, x_n$ and the lattice of integer points defined by the parametric variables. Among other things, this assures that whenever the parametric variables have integer values so will the original variable have and vice versa. This one-to-one correspondence can be proved by showing that the determinant of the matrix D that expresses the x_i's in terms of the parametric variables is always +1 or -1 (D is formed with the n rows labeled $x_1, \ldots, x_n$ plus an auxiliary row $-1 = -1$ that helps to complete a $(n + 1)$ by $(n + 1)$ matrix. We will initially have $|D| = \pm 1$, and our pivoting formulas result in column operations that do not change the absolute value of $|D|$.)

In geometric terms, whenever there is a breakthrough in a certain column, this implies that the intersection of the $n - 1$ planes correspond to the other parametric variables, coincides with one of the edges of the restricted polyhedron.

Geometric visualizations of some of the aspects of the problem, although difficult to accomplish, provide additional insight into the workin

of the algorithm.

An Important Observation about Gomory Cuts

Another important observation and one that, to the best of our knowledge, has not explicitly been made by Gomory or by anybody else, is the following:

Any Gomory cut generated from a certain row i (constraint i) will be such that it will be satisfied by every non-negative integer point (non-negative with respect to the current set of parametric variables) that satisfies constraint i . However, the Gomory cut can be, and most often will be, unsatisfied by some feasible integer points (feasible with respect to constraint i) that are non-negative wit . respect to the original variables. The Gomory cut generated from row i is then also a function of the rows that have been used to generate the current set of parametric variables.

6.3 Inverse Pivoting

By successive pivoting operations we express the original variables $(x_1, \ldots, x_n)$ in terms of the parametric variables. If we wanted to express the parametric variables in terms of the original variables, we would have to invert the matrix that expresses the previous relationship. However, if we choose to carry this inverse matrix along with our regular tableau, the updating operation is a simple one, and there is the added advantage of having obtained the inverse for every intermediate tableau and not just for some specific tableau. The updating operation will be called inverse pivoting, and it will be defined in the following paragraphs.

Let us establish a matrix notation that is slightly different from th of (2.1.5):

Let

$$s = \begin{bmatrix} z \\ s_1 \\ \cdot \\ \cdot \\ \cdot \\ s_m \end{bmatrix} \quad A^k = \begin{bmatrix} a^k_{00} & a^k_{01} & \cdots & a^k_{0n} \\ a^k_{10} & & & \cdot \\ \cdot & & & \cdot \\ \cdot & & & \cdot \\ \cdot & & & \cdot \\ a^k_{m0} & & & a^k_{mn} \end{bmatrix} \qquad (6.3.1$$

$$x = \begin{bmatrix} -1 \\ x_1 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix} \quad D^o = \begin{bmatrix} -1 & & & & & \\ & -1 & & & & \\ & & \cdot & & & \\ & & & \cdot & & \\ & & & & \cdot & \\ & & & & & -1 \end{bmatrix} \quad T^k = \begin{bmatrix} 1 \\ -t^k_1 \\ \cdot \\ \cdot \\ \cdot \\ -t^k_n \end{bmatrix}$$

We will have

$$\begin{bmatrix} s \\ x \end{bmatrix} = \begin{bmatrix} A^o \\ D^o \end{bmatrix} \quad T^o = \begin{bmatrix} A^k \\ D^k \end{bmatrix} \qquad (6.3.2)$$

and

$$\begin{bmatrix} A^{k+1} \\ D^{k+1} \end{bmatrix} = \begin{bmatrix} A^{k} \\ D^{k} \end{bmatrix} \quad P^k \qquad (6.3.3)$$

The square matrix P^k is the transformation matrix that expresses the pivoting operation. Because we pivot on the $j0^{th}$ element of the row $(b_0^k, b_1^k, \ldots, b_n^k)$, and because $b_{j0}^k = 1$, we have the following:

$$P^k = \begin{bmatrix} 1 & & & & & & & & \\ & \cdot & & & & & & & \\ & & \cdot & & & & & & \\ & & & 1 & & & & & \\ -b_0^k & \cdot & \cdot & -b_{j0-1}^k & -1 & -b_{j0+1}^k & \cdot & \cdot & -b_n^k \\ & & & & & 1 & & & \\ & & & & & & \cdot & & \\ & & & & & & & \cdot & \\ & & & & & & & & 1 \end{bmatrix} \qquad (6.3.4)$$

From (6.3.3) we have

$$D^2 = D^1 P^1 = D^o P^o P^1$$

using (6.3.2) we obtain, in general

$$X = D^o P^o P^1 \ldots P^{k-1} T^k$$

from which

$$T^k = (D^o P^o P^1 . . . P^{k-1})^{-1} \ X = (P^{k-1})^{-1} . . . (P^1)^{-1} (P^o)^{-1} (D^o)^{-1} ;$$

if we define $E^o = (D^o)^{-1} = D^o = - I$ (6. 3. 6)

and

$$E^{k+1} = (P^k)^{-1} \ E^k , \quad k = 0, 1, . . . \tag{6.3.7}$$

then

$$T^k = E^k X \tag{6.3.8}$$

Matrix P^k is an <u>elementary</u> matrix, and because row j0 has a - 1 in the diagonal, it turns out that <u>it is its own inverse</u>, as can be easily verified by multiplication. Equation (6. 3. 7) can then be expressed as

$$\left.\begin{aligned} e^{k+1}_{ij} &= e^k_{ij} \ ; \quad \text{for} \ i \neq j0 \\ e^{k+1}_{j0,j} &= - \sum_{i=0}^{n} b^k_j \ e^k_{j0,j} \ ; \end{aligned}\right\} \quad j = 0, 1, . . . , n \tag{6.3.9}$$

We may carry matrix E^k in our computations and perform the addition of rows expressed in (6. 3. 9), but it is mnemonically easier to carry its transposition and to perform the operation on the corresponding columns. Thus, under each column of the regular tableau we will have another column that expresses the corresponding parametric variable in terms of the original variables.

This technique of inverse pivoting is specially useful for research or whenever one is interested in learning how the successive Gomory-cuts look like in the original space.

6.4 Natural Inequalities

Let us consider the following inequality:

$$8x_1 - 7x_2 + 5x_3 \leq 19 \qquad (6.4.1)$$

Where x_1 , x_2 and x_3 are non-negative integer variables. If we let $x_2 = x_3 = 0$ then (6.4.1) would imply $x_1 \leq 2 \leq [19/8]$. Similarly if $x_1 = x_2 = 0$ w would have $x_3 \leq 3 \leq [19/5]$. On the other hand, if we generate a Gomory-cut from (6.4.1) with $p = 8$ and another one with $p = 5$, we will have:

With $p = 8$: $$x_1 - x_2 \leq 2 \qquad (6.4.2)$$

With $p = 5$: $$x_1 - 2x_2 + x_3 \leq 3 \qquad (6.4.3)$$

We see that (6.4.2) strictly binds x_1 (assuming $x_2 = x_3 = 0$) to the integer value that (6.4.1) only implies; and (6.4.3) binds x_3 in a similar way. We say that (6.4.2) is a natural inequality of (6.4.1) for x_1 ; and (6.4.3) is a natural inequality of (6.4.1) for x_3 .

In general, if we have an inequality:

$$a_1 x + a_2 x_2 + \ldots + a_n x_n \leq a_0 \, ; \qquad (6.4.4)$$

and if a_1 has the same sign as a_0, then the Gomory-cut generated from (6.4.4) with $p = |a_i|$ is said to be the natural inequality of (6.4.4) with respect to x_i.

All the Gomory-cuts that are used in our primal algorithm are natural inequalities. (They are "zero natural inequalities" or "non-zero natural inequalities" according to whether they are "zero-cut" or not.)

6.5 Deepest Cuts and Hierarchical Rules

We define a deepest cut as any of the hyperplanes that correspond to the several faces of the restricted polyhedron. A deepest cut in this sense, is different than the deepest cuts mentioned in [5], [8] and [11], where the concept is more that of "deepest" in a certain direction.

The number of faces of the restricted polyhedron may be smaller than, equal to, or larger than the number of faces of the original polyhedro as may be verified by drawing simple two dimensional examples.

A sufficient, but not necessary, condition for a certain parametric variable t_j to be a "deepest-cut variable" is if there is a breakthrough in every one of the other columns. This is because it shows that, besides the current vertex, there are other $n - 1$ vertices of the restricted polyhedron that lie on the hyperplane $t_j = 0$, thus determining it as a deepest cut.

In general, it is difficult to have a tableau in which every parametric variable is a deepest cut variable. This is because by using an "all-integer" method, we are limiting ourselves in our choice of sets of parametric variables. (If we take any n deepest cuts that define a vertex of the restricted polyhedron and if we were to use this set as a system of coordinates to express the original constraints, we would have to use, in general, fractional coefficients.)

However, it is conceivable that one of the parametric variables (say t_n) might correspond to a deepest cut; and that the projection of a second one (say t_{n-1}), might be a deepest cut of the (n-1) dimensional projection of the restricted polyhedron over the hyperplane $t_n = 0$; and subsequently, t_{n-2} could be a (n-2) dimensional deepest cut, and so forth. This is the motivation for the hierarchical rules that were given in section 4.3. A deeper study of this aspect could lead to better hierarchical rules.

6.6 Geometric Considerations of Two-Dimensional Convergence

In section 4.2 we developed a proof of finite convergence of the basic algorithm when applied to two-dimensional problems. In this proof, an algebraic concept of dominance of rows was used. We will now present the geometric motivation of this concept.

Let us consider the two-dimensional space as given in terms of the current parametric variables t_a and t_b. We will define the <u>primary region</u> as the region given by $t_a \geq 0$, $t_b \geq 0$, $z \geq a_{0,0}$. That is, every point in this region is non-negative and would give z a value not smaller than its current value. Clearly, the optimal solution, (assuming it exists), is to be found in the primary region.

If a certain constraint (row) is satisfied by every point of the primary region, the constraint is said to be <u>dominated;</u> if it is not satisfied by a subset of this region it is said to be <u>non-dominated.</u> This geometric definition of dominance coincides with the algebraic definition given in section 4.2, as it can easily be shown.

When the primary region does not include any point of the positive orthant (i.e. any point such that $t_a > 0$, $t_b > 0$) it means that the objective function, z, has reached its optimal value, and the present solution is optimal (although there might be other optimal solutions).

Whenever we generate a cut for pivoting, it will be non-dominated and, once we pivot, the cut will become one of coordinate axis and the new primary region will be a subset of the previous one. This will occur whether we have a zero- or a non-zero-cut. If we have a non-zero-cut, there will be a shift of the line $z = a^k_{0,0}$ (when $a^{k+1}_{0,0} > a^k_{0,0}$) so that the primary region will be reduced further.

What the proof of section 4.2 shows, is, in geometric terms, that as we introduce new cuts, the primary region diminishes consistently until the positive orthant (in terms of the current set of parametric variables) is completely excluded. At this point optimality is proved.

In other words, every time we pivot, the system of coordinates is translated from one vertex to another and/or rotated around a vertex of the restricted polyhedron, so that eventually there is no better point in the non-negative orthant of the (current) system of coordinates.

CHAPTER 7

SUGGESTED EXTENSIONS

In the first section of this chapter we present the concept of a "Tight" Algorithm which would be similar to our basic algorithm except that the parametric variables would always correspond to deepest cuts. In the second section we introduce the notion of fractionalization of variables, and we advance some ideas as to how this concept could be used to develop a new approach for solving mixed integer LP problems.

7.1 A "Tight" Algorithm

Further study of the Gomory cuts in the context of the primal feasible algorithm should lead to a procedure for generating some of the deepest cuts that define any specific vertex of the restricted polyhedron. By using inverse pivoting these deepest cuts may be expressed in terms of some initial set of parametric variables.

To be more specific, let us assume that we have our tableau (with an initial feasible solution), and that every parametric variable t_j is a deepest cut variable. This means that there is a breakthrough in every column (some are in the maximizing direction, some in the minimizing direction). If we are maximizing we could choose any negative column and we would be able to obtain a non-zero-cut that, upon pivoting, would take us to another vertex. This non-zero-cut will not necessarily be a deepest cut. However, we might be able to find a "deepest cut procedure" so that, in fact, we obtain the expression of that deepest cut which would not only move to the next vertex, but the tableau would again be entirely in terms of deepest cut variables.

The tableau would have rational coefficients, rather than integer ones, but by appropriately changing the scale of some or all of the row variables, we should be able to keep the new coefficients in terms of integers.

A "Tight" Algorithm could have two special advantages: in the first place, it might turn out to be computationally more efficient than the Basic Algorithm (i.e. it might be an "accelerated" version of the Basic Algorithm); in the second place, the fact that at every vertex the constraints may be expressed in terms of a uniquely defined set of parametric variables (i.e. the variables corresponding to the deepest cuts that define that vertex) could turn out to have important consequences for the study of duality in integer LP.

7.2 Fractionalization of Variables and Mixed Integer LP

Two of the reasons for the use of all-integer algorithms in integer LP are: first, to obtain exact solutions; and second, to use, at least implicitly, the number theoretic properties of the coefficients. However, it should be pointed out that the generation of Gomory cuts is valid whether the coefficients are integer or not. For example, if all the coefficients of the tableau (excepting the last n rows which give the integer variables x_j, $j = 1, \ldots, n$ in terms of the parametric variables t_j) were not integral, our primal feasible algorithm should still work. The slack variables might not have integer values, but the x_j's will always have integer values.

We define fractionalization of a variable as the process of replacing a parametric variable t_j by another variable t'_j such that $t_j = t'_j/k$, where k is some positive integer. This is equivalent to dividing every element in column j by k. We could either express these new elements as rational numbers, or as decimal fractions; or we could multiply every element of the tableau by k, thus eliminating fractions by changing the scale of the row variables.

The result of this fractionalization is that the lattice of integer points of the new system of coordinates is "finer" (in the t_j-direction) than the previous lattice. However, every point of the old lattice is also a point of the new one.

If all the coefficients and constants of a mixed integer LP problem are rational numbers, then every extreme ›lution (i.e. every vertex of the restricted polyhedron) will be in terms of rational numbers. It is conceivable that a mixed integer LP problem could be transformed into a pure integer LP problem by proper fractionalization of some of the variables.

BIBLIOGRAPHY

I - INTEGER LINEAR PROGRAMMING

[1] BEN-ISRAEL and CHARNES, A. - "On Some Problems of Diophantine Programming", Cahiers du Centre d'Etudes de Recherche Opérationnelle, pp. 215-280, 1962.

[2] GLOVER, FRED - "A Study of the All-Integer Integer Programming Algorithm", O.N.R. Research Memorandum No. 116, Carnegie Institute of Technology, 1963.

[3] GLOVER, FRED - "A Bound Escalation Method for the Solution of Integer Linear Programs", O.N.R. Research Memorandum No. 119, Carnegie Institute of Technology, 1963.

[4] GOMORY, R. E. - "Outline of an Algorithm for Integer Solutions to Linear Programs", American Math. Soc. Bulletin, Vol. 64, No. 5, 1958.

[5] GOMORY, R. E. - "An Algorithm for Integer Solutions to Linear Programs", Technical Report No. 1, (Princeton - I.B.M. Mathematics Research Project), Nov. 17, 1958. (Also in Ref. [7], pp. 269-302).

[6] GOMORY, R. E. - "An All-Integer Integer Linear Programming Algorithm", I.B.M. Research Report, RC-189, Jan. 29, 1960. (Also as Chapter 13 of Ref. [9]).

[7] GRAVES, R. L. and WOLFE, P. (editors) - Recent Advances in Mathematical Programming, McGraw-Hill Book Co., 1963.

[8] MARTIN, GLENN T. - "An Accelerated Euclidean Algorithm for Integer Linear Programming", Recent Advances in Mathematical Programming, Graves and Wolfe, editors, McGraw-Hill Book Co., pp. 311-317, 1963.

[9] MUTH, J. F. and THOMPSON, G. L. (editors) - Industrial Scheduling, Prentice-Hall, Englewood Cliffs, N. J., 1963.

[10] THOMPSON, G. L. - "The Stopped Simplex Method: I. Basic Theory for Mixed Integer Programming; Integer Programming", Revue Francaise de Recherche Opérationelle, Vol. 8, No. 31, pp. 159-182, 1964.

[11] VAN SLYKE, RICHARD and WETS, ROGER - "On Diagonalization Methods in Integer Programming", O. R. Center Research Report No. 27, Univ. of Calif., Berkeley, pp. 25, (U.S. Govt.: Doc. No. AD-288053), 1963.

II - MIXED INTEGER LINEAR PROGRAMMING

[12] BALAS, EGON - "Un Algorithme Additif pour la Résolution des Programmes Linéares en Variables Bivalentes", C.R. Acad. Sci. Paris, 258 (1964), 3817-3820.

[13] BEALE, E. M. L. - "A Method of Solving Linear Programming Problems When Some but Not All of the Variables Must Take Integral Values", Technical Report No. 19, (Princeton University Statistical Techniques Research Group), July 1958.

[14] BENDERS, J. F. - "Partitioning Procedures for Solving Mixed-Variables Programming Problems", Numerische Mathematik, Vol. 4, pp. 238-252, 1962-63.

[15] FEDEROWICZ, ALEXANDER V. - "A Generalized Algorithm Solution of a Class of Non-Convex Programming Problems", Ph.D. Thesis in Mathematics, Carnegie Institute of Technology, 1963.

[16] GOMORY, R. E. - "A Method for the Mixed Integer Problem", The RAND Corporation, Santa Monica, (RM-2597), 1960.

[17] HARRIS, PAULA M. V. - "An Algorithm for Solving Mixed Integer Linear Programmes", O.R.Q., Vol. 15, No. 2, pp, 117-132, 1964.

[18] HEALY, W. C., JR. - "Multiple Choice Programming" (A Procedure for Linear Programming with Zero-One Variables), Operations Research, Vol. 12, No. 1, pp. 122-138, 1964.

[19] LAND, A. H. and DOIG, A. - "An Automatic Method of Solving Discrete Programming Problems", Econometrica, Vol, 28, pp. 497-520, 1960.

[20] SZWARC, WLODZIMIERZ - "The Mixed Integer Linear Programming Problem When the Integer Variables are Zero or One", Carnegie Institute of Technology, 1963.

[21] TONGE, FRED M. - "A Revised Algorithm for the Mixed Integer Programming Problem with Boolean Variables", Carnegie Institute of Technology, Research Memorandum for Private Circulation, June 1963.

III - NUMBER THEORY

[22] BLANKINSHIP, W. A. - "A New Version of the Euclidean Algorithm", American Mathematical Monthly, 70, pp. 742-745, 1963.

[23] ORE, O. - Number Theory and Its History, New York, McGraw-Hill Book Co., 1948.

[24] USPENSKY, J. V. and HEASLET, M. H. - Elementary Number Theory, New York, McGraw-Hill Book Co., 1939.

APPENDIX A

SOME CONSIDERATIONS ABOUT THE LEXICOGRAPHICAL SIMPLEX METHOD

A. 1 Proof of Optimality

Lemma A. 1 If the tableau is both primal feasible and dual feasible, the current solution is optimal.

Proof: Let us assume that we are maximizing. If the tableau is dual feasible, it then means that $a^k_{0,j} \geq 0$; $j = 1, \ldots, n$. The current expression for z is:

$$z = a^k_{0,0} + \sum_{j=1}^{n} a^k_{0,j} (-t^k_j)$$

We can restrict the t^k_j's to be non-negative without losing any feasible (integer) solution. But if the t^k_j's are non-negative, this means that the largest value of z is precisely $a^k_{0,0}$, and therefore, the current solution, being feasible, is also optimal.

If we were minimizing, dual feasibility would imply that $a^k_{0,j} \leq 0$; $j = 1, \ldots, n$ and the proof would follow in a similar way. (Q. E. D.)

A. 2 Proof of Lexico-Optimality

Lemma A. 2 If the tableau is both primal feasible and lexico-dual feasible, the current solution is lexico-optimal.

Proof: Let us assume that we are lexico-maximizing. If the tablea is lexico-dual feasible, it means that every column (not considering column zero) is lexico-positive; that is, the first non-zero element of each column is positive. This means that $a_{0,j} \geqslant 0$, $j = 1, \ldots, n$, and from lemma A. 1, we realize that z has attained its maximum value. We have to check whether the solution is the lexico-largest feasible solution. That is, we have to see if the h^{th} component of the solution could increas its current value without decreasing the value of any one of the previous components. Consider the following row:

$$s_h = a^k_{h,0} + \sum_{j=1}^{n} a^k_{h,j} (-t^k_j)$$

Now, because we constrain the t^k_j's to be non-negative, s_h can only increase if $a^k_{h,j} < 0$ for some j, say $j = j0$. But if $a^k_{h,j0} < 0$, this means that there must be some $i < h$ such that $a^k_{i,j0} > 0$, because every column is lexico-positive. Thus, if we increase t_{j0}, the value of s_h will certainly increase, but s_i will decrease, and this will result in a lexico-smaller solution. Therefore, the current solution is the lexico-largest.

A similar argument would be used if we were lexico-minimizing.

(Q. E. D.)

A. 3 The Usefulness of a Lexicographical Simplex Method

If we are minimizing with a "plain" simplex method, we consider for pivoting only those columns j with $a_{0,j} > 0$; whereas, if we are using a lexicographical method, we will consider those columns which are lexico-positive.

Let us consider the following problems:

$$\begin{aligned} \min z = \quad & x_2 - x_3 \\ \text{with} \quad -x_1 - x_2 + 2\,x_3 &\leq 1 \\ x_1 - 2\,x_2 + x_3 &\leq 1 \\ x_1, x_2, x_3 \ \text{non-negative integers} \end{aligned}$$

If we try to solve this example with our basic algorithm, and using a "plain" criterion for considering pivoting columns, we will always have a single choice of pivoting column and a single choice of generating row. Furthermore, cycling will occur and the optimal solution will not be found.

On the other hand, if we use a lexicographical criterion for considering pivoting columns, we will obtain the lexico-optimal solution in four iterations. This example shows that the lexicographical simplex method, or some other device to stop cycling, is necessary for our bas

algorithm.

APPENDIX B

LEXICO-MINIMIZATION AS A PROCEDURE FOR IMPLICIT TABLEAU REDUCTION

In Chapter 3 it was shown how one could take advantage of any equalities in the original problem, for reducing the size of the tableau through the elimination of zero-variables. It turns out that an equivalent result can be obtained in some instances by properly rearranging rows and then lexico-minimizing the problem.

More specifically, let us assume that we have a tableau such that the only infeasibilities correspond to some zero-variables that have non-zero values. We now rearrange the rows so that all the zero-variable rows occupy the first places (including those zero-variable rows that are feasible), followed by the objective variable row, by the rows corresponding to non-negative variables, and by the rows corresponding to other unsigned variables (we may have or may not have an objective variable).

If the problem is lexico-minimized and there exists a feasible solution, the first components of the solution - those corresponding to the zero-variables - will have a value of zero. Furthermore, because the lexico-optimal solution requires that every column be lexico-negative, this implies that there is no positive element in the first row. As a matter of fact, in the process of solution there will be a step where only one negative element will be left in row 1; every other element being zero. It follows that the column corresponding to this negative element will never again be chosen for pivoting because it is lexico-negative and there is no way in which it can change its lexicographical sign. This is what we call implicit reduction of the tableau. Similarly, the second row will reach a later step where (if we leave out the column implicitly reduced by row 1) it will have a single negative element among zero elements. This negative element determines another negative column that is implicitly reduced, and which will not be chosen for pivoting. A similar thing may happen with every zero-variable row. The net result is that the tableau automatically acquires a triangular canonical form.

BIOGRAPHICAL NOTE

Rómulo Hector González-Zubieta was born in Mexico City, Mexico, on September 11, 1939. He attended private primary and secondary schools in Monterrey, Mexico.

He attended the Instituto Tecnologico de Monterrey and received the degrees of Licenciado en Matematicas and of Licenciado en Fisica, both with "Mencion Honorifica", in June, 1960.

Mr. González-Zubieta received a U. S. State Department Scholarship through the Institute of International Education for study at the Massachusetts Institute of Technology in 1960-1961. He has been appointed Research Assistant at the Operations Research Center, M.I.T. during the four academic years 1961-1965. The Banco de Mexico, S.A. has granted him financial support from September, 1963 to June, 1965. In June, 1962, Mr. González-Zubieta received the degree of Master of Science in Industrial Management. His thesis was awarded the E. P. Brooks Prize for the best graduate thesis in the School of Industrial Management in 1962.

During the summers of 1961, 1962 and 1963 he has been a consultant for IBM de Mexico, Empaques de Carton Titan, and C-E-I-R de Mexico, respectively.

Mr. González-Zubieta is a member of the Operations Research Society of America, and of The Institute of Management Sciences.

In December 1963, he married the former Bertha Diaz Barreiro Saavedra. Their first child is Monica Irene.